AMIGA WELT

5/88 November/Dezember DM 12,–

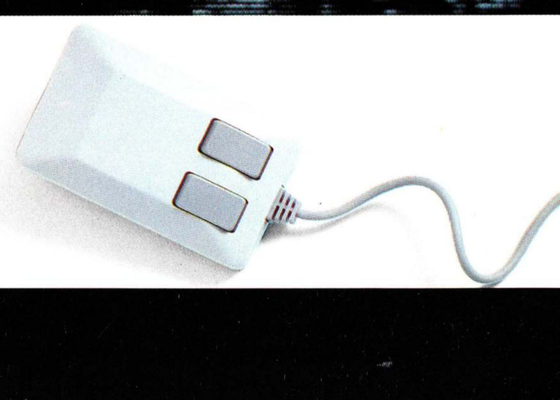
Video-Grafik
für jedermann

Marktübersicht, Public-Domain, Tips

Neue Runde im DOS-Streit
ARP statt 1.3



15 Seiten Spielehits
Larry macht alle an



Kurs: C als 1. Fremdsprache

sfr. 12,–; ÖS 95,–; Fmk 41,–; hfl 15,–; Lire 12500; lfr. 300; ISSN 0933-8616

Apropos ... Vermutungen



Manfred S. Schmidt (Chefredakteur)

Hurenbock Larry (Besprechung auf Seite 138), in Amerika bereits Kultfigur, hat Einzug in mitteleuropäische Zeitgeist-Magazine gehalten. Wohlwollende Rezensionen waren ihm sicher. Jetzt hat er Probleme. Nicht mit dem Virus, aber mit Viren. Software-Importeure legen Wert auf die Feststellung, daß das Sex-Abenteuerspiel im Original virenfrei sei. Deshalb liegt die Vermutung nahe, daß von dritter Seite kopierten Versionen allerlei Kapriolen eingeimpft wurden. Die Nutzung kopierter Software wird damit mehr und mehr zum unkalkulierbaren Risiko. Es soll inzwischen Software-Häuser geben, die auf jeglichen Kopierschutz verzichten und statt dessen ein Strafprogramm ins Original pflanzen. Und das wiederum wird vermutlich ein gutes Geschäft für all diejenigen, die Anti-Virenprogramme anbieten.

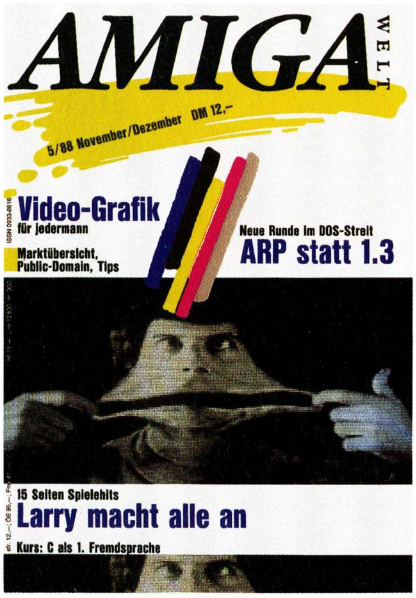
Warum wurde die auf der für Anfang September in Innsbruck stattfindenden Händler-Tagung vorgesehene Vorstellung der Workbench 1.3 gecancelt? Commodores Geschäftsführer in Österreich, Wilhelm Draxler: „Weil noch Fehler gefunden wurden." Das gibt Anlaß zu weiteren Vermutungen, denn bislang hat jede Version eines Betriebssystems Fehler aufgewiesen: Erinnern wir uns an die verschiedenen MS-DOS-Versionen (1.2, 3.2, 3.3 bis 4.0). Wenn es keine Nachbesserungsmöglichkeiten gäbe, gäbe es auch keine verbesserten Versionen. Das läßt drei weitere Vermutungen zu. Vermutung Eins ist eine auch von Commodore angedeutete: 1.3 kommt deshalb nicht, da die Version 1.4 in Vorbereitung ist. Dagegen spricht allerdings, daß sich auf der 1.4 nur Routinen für die neuen Grafik-Chips und den Hedley-Monitor befinden werden.

Diese Neuheiten werden aber in der Produktpalette von Commodore vorläufig noch nicht auftauchen. Warum sollen die User also warten? Vermutung Zwei geht davon aus, daß Commodore inzwischen darüber verärgert ist, daß bereits viele WB 1.3-Disketten zirkulieren, und versuchen will, an der offiziellen Version zu verdienen. Dagegen spricht allerdings, daß sich die in Aussicht gestellte offizielle Version genauso in Windeseile verteilen wird, da Workbench-Disketten nicht kopiergeschützt sind. Dazu kommt, daß eine absichtlich herbeigeführte Verwirrung nicht im Interesse eines Hardware-Herstellers sein kann. Die dritte Vermutung geht dahin, daß Commodore sich noch in Verhandlungen über die Urheberrechte einiger Programmteile befindet. Für diese Annahme spricht, daß zahlreiche Programmteile aus der Public-Domain-Ecke kommen, und denkbar ist auch, daß die Verhandlungen mit Microsmith, die das „Amiga-DOS Replacement Project" betreiben, in eine erfolgsversprechende Phase gekommen ist. Die vorhandenen ARP-Befehle jedenfalls sind durchwegs effizienter als ihre 1.3 Gegenstücke (siehe „Trimmplan für Amiga-DOS" auf Seite 83).



Uwe Knierim (Redakteur)

Inhalt



Grafiken und Animationen sind Amigas Spezialgebiet. Diese Talente lassen sich auch von Videofilmern nutzen.

Notepad

Neuigkeiten aus der Computerszene 6

Amigawelt

Vom Computertraum zum sendefähigen Film
In einem Kölner Fernsehstudio hat sich der Amiga unentbehrlich gemacht 14

Kinnhaken mit Dynamic Drums
Bei einer Stuntmanshow brillierte der 500er in ungewöhnlicher Umgebung 18

Amigakicks

Kurze, hilfreiche Tips von Lesern 62

Hard- und Software

Dämonisches aus den Tiefen des Raumes
Ein Editor für alle 3D-Programme will 3-Demon sein 24

Dreimal Vielseitigkeit zum Thema Video
Spezialhardware für besondere Video-Effekte 26

Turboprint optimiert den Drucker 28

Einkaufsliste für Videografen
Alles, was Computerfilmer benötigen, findet sich in dieser großen Marktübersicht 30

Fernsehstudio in Deluxe-Qualität
Videopräsentationen werden mit diesem neuen Produkt von Electronic Arts aufgewertet 40

Soundtool für Spieleprogrammierer 42

Sprechblasenhelden erobern den Bildschirm
„Comic Setter“ ist endlich realisiert 44

Drei auf der Straße nach Modula
In einem Vergleichstest beweisen sich Varianten der Programmiersprache Modula auf rauer Piste 47

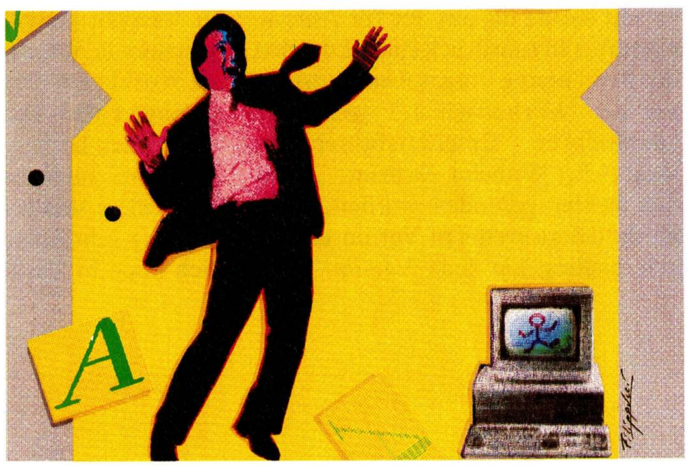
Public Domain

PD-Software — Fundgrube für Desktop-Video-Anwender 50





Selbst in der gefährlichen Welt der Stuntmen kann Amiga seine Dienste anbieten.



Was es alles an Hard- und Software für Videografen gibt, zeigt unsere Einkaufsliste.

Fischers Fritze fischt frische . . .
Fred Fishs neue Ausbeute wird untersucht 58

DME — das Chamäleon unter den Editoren
Der Public-Domain-Editor wird ausführlich dokumentiert 60



Know How

C als erste Fremdsprache, 1. Teil
Wer um- oder aufsteigen möchte, findet hier Anschluß 64

PAL — oder wie die Farbe auf den Bildschirm kommt 68

Einstieg mit BASIC
Im zweiten Teil des Kurses werden einige einfache Runden gedreht 70

Hollywood im Arbeitszimmer
Überraschende Effekte lassen sich auch ohne teures Zubehör aufs Videoband zaubern 74

Computermalschule, Teil 5
Strahlen machen elastisch — wieso, zeigt Joel Hagen 78

Trimmplan für AmigaDOS
ARP macht der Workbench 1.3 Konkurrenz 83

Schlagkräftige Devicetruppe dank Workbench 1.3
AUX:, PIPE: und RAD: erweitern das Repertoire an Devices 88

Listing

Abtippen jetzt noch leichter mit checksum
Die neue Version des Checkers gibt es auch in BASIC 94

Welche Preferences hätten'S denn gern?
Verschiedene System-Einstellungen auf Abruf 102

KeyMaps — Zeichen setzen nach Bedarf
Nicht nur eine HEX-Tastatur, sondern auch reichlich Grundlagen über das Keyboard vermittelt dieses Listing 106

Mit BASIC-Testbild zum optimalen Bildschirmfoto 121

2000-Mark-Superlisting:
Universelle Konvertierungshilfe für Daten, Texte und Programmfiles 122

Automatische Fehlersuche mit Memwatch 126

Spiele

Spieletests und News 130

Larry auf Minnepfaden
Anmachen ist nicht einfach, wenn man immer nur vor dem Bildschirm gehockt hat 138

Tips zu „Leisure Suit Larry in the Lounge Lizards" 141

Auf den Spuren von „Obliterator" 142

Rubriken

Editorial 3

Diskettenservice: siehe unter Notepad auf Seite 10

Bücher 20

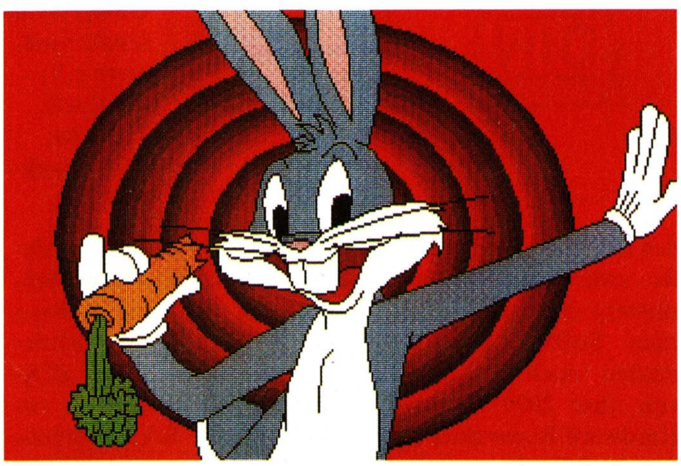
Leserbriefe 37

Infomarkt 117

Kleinanzeigen 120

Impressum/Inserentenverzeichnis 133

Vorschau 146



Für Video-Desktop-Anwendungen ist das Public-Domain-Angebot eine Schatzkiste.



Mit abgeschaltetem Motor schlich sich eine U-Boot-Simulation auf die Indizierungs-Liste.

Hitex-Editor V2.0, der Mischling

Jetzt gibt es einen kompakten, schnellen und leistungsfähigen Editor für den Amiga. Der Standardeditor „Ed“ gilt als umständlich in der Handhabung und zu langsam. Die bisher einzige Alternative war der Micro-Emacs-Editor, der auf der Extras-D-Diskette zu finden ist. Dieser arbeitet zwar flott, beansprucht aber sehr viel Hauptspeicher (zirka 60 KByte), da er in C programmiert ist.

Der Hitex-Editor vereint die positiven Eigenschaften folgender Editoren:

- FSE/ASM vom Commodore 64,
- Editor des Apple-Pascal-Systems vom Apple II,
- Tempus vom Atari-ST,
- Ed vom Commodore-Amiga,
- Editor des Amiga-BASIC,
- Micro-Emacs vom Commodore-Amiga.

Daraus ergibt sich ein benutzerfreundlicher, kompakter und äußerst schneller Hochleistungseditor. Er wurde vollständig in M68000-Assembler entwickelt, woraus sich die hohe Arbeitsgeschwindigkeit ergibt. Mit dem Hitex-Editor lassen sich problemlos Programme in sämtlichen Sprachen realisieren. Praktisch ist auch, daß er sowohl vom CLI, als auch von der Workbench aus gestartet werden kann. Erfolgte der Start von der Workbench aus, so erzeugt der Editor automatisch zu jeder abgespeicherten Datei ein Icon. Der Cursor kann mit den Tasten, aber auch mit der Maus positioniert werden. Sämtliche Befehle werden über eine Menüleiste erreicht, die wichtigsten auch über Tastatur, was sehr zur Benutzerfreundlichkeit des Editors beiträgt.

Eine Bildschirmseite besteht aus 80 Zeichen pro Zeile und 30 beziehungsweise 60 (im Interlace-Modus) Textzeilen. Die maximale Zeilenlänge beträgt 254 Zeichen, wobei leider nur die ersten 160 editierbar sind. Weiterhin hat man die Möglichkeit, zehn Files gleichzeitig zu bearbeiten, die in Puffern abgelegt werden. Eine sehr leistungsfähige Batch-Sprache, die durch Macros definiert wird, ermöglicht eine komfortable Textbearbeitung. Blöcke werden mittels Cursor oder Maus markiert.

Ein markierter Block ist permanent sichtbar und kann mit umfangreichen Blockoperationen bearbeitet werden. Zusätzlich besteht die Möglichkeit, 20 Sprungmarken zu setzen. Beeindruckend sind die schnellen Such- und Ersetzoperationen, bei denen mit Wildcards gearbeitet werden kann. An der rechten Seite des Textausschnitt-Fensters befindet sich ein Scrollbar, mit dessen Hilfe ein komfortables Textscrolling ermöglicht wird. Die unterste Zeile des Screens dient als Kommandozeile und wird zur Kommunikation mit dem Editor benützt. Der Hitex-Editor besitzt eine automatische Einrückfunktion, mit der bequem in blockorientierten Sprachen wie C oder Pascal programmiert werden kann.

Die technischen Daten des Hitex-Editors:

- Laden von 47 000 Bytes mit 2700 Zeilen: 6,5 Sekunden,
- Bildschirmaufbau im Normalmodus (30 Textzeilen): zirka 0,037 Sekunden,
- Scrolling im Normalmodus: zirka 93 Zeilen pro Sekunde,
- Suchen und Ersetzen: zirka 2000 Ersetzungen pro Sekunde, maximal 5600 Ersetzungen pro Sekunde,
- Länge des Programms: zirka 34 KBytes.

(Thomas Meidinger)

Info: Dietmar Heidrich, Blenderstraße 60, 2800 Bremen 61, Preis: DM 79,—.

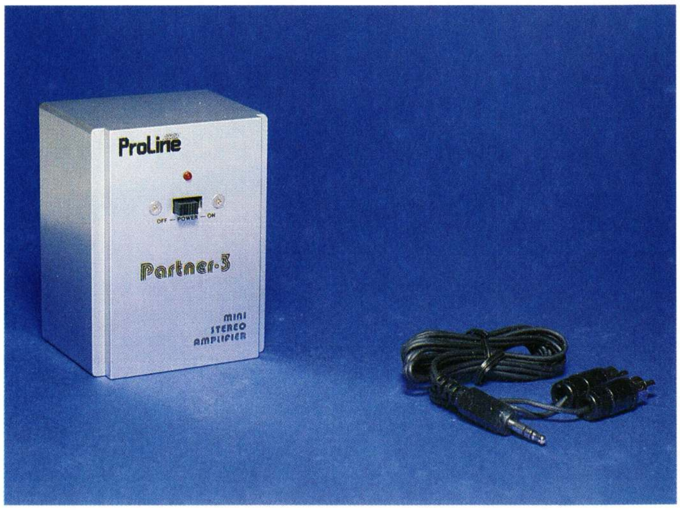
TÜV gegen Softschrott

114 000 Mark Schaden hat defekte Software bei einem Immobilienmakler angerichtet. Ist der — aus verständlichen Gründen nicht namentlich genannte — Rekordhalter zwar eine Ausnahme, entsteht dem durchschnittlichen Software-Opfer immerhin noch ein Schaden von über 2000 Mark. Für Hardware-Reparaturen werden im Mittel lediglich 320 Mark aufgewendet. Das ergab eine Umfrage, die TÜV und Chip im vergangenen Herbst starteten. Bei dieser Aktion unter dem Thema „Softschrott“ wandte man sich an kommerzielle PC-Anwender, die schlechte Erfahrung mit fehlerhaften Programmen gemacht hatten. 44 Prozent von knapp 900 Einsendern beanstandeten Textverarbeitungen, die zu den am weitesten verbreiteten Anwendungen zählen. Integrierte Programmpakete zeigten, gemessen an ihrer geringen Verbreitung, am meisten Schwachstellen.

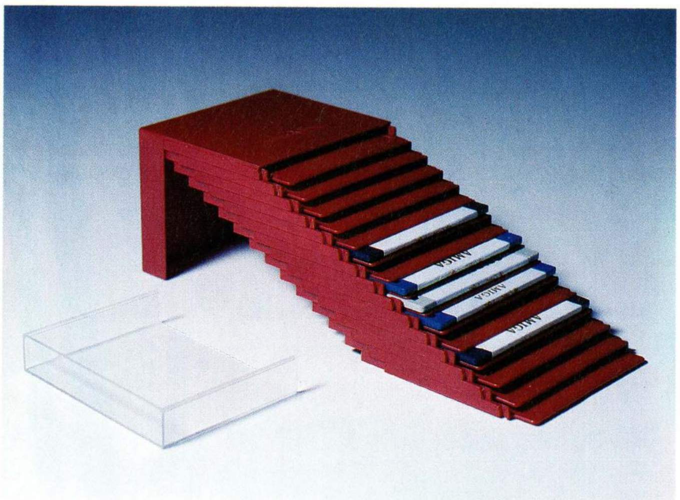
„Weder bekannte Hersteller noch Standardtitel garantieren Qualität,“ stellt Rainer Korbmann, Chefredakteur der Chip fest, „Software wird immer komplexer und damit fehleranfälliger“. Geprellten Anwendern ist dies ein schwacher Trost. Sie sind eine Gruppe ohne Lobby und stehen Herstellern und Verkäufern hilflos gegenüber. Im Frühjahr 1985 wurde der TÜV dagegen initiativ. Experten des Überwachungsvereins boten eine Prüfplakette für bestandenen Softwaretest an. Meßlatte für die Untersuchung ist die Deutsche Industrienorm. Neben dem DIN-Qualitätssiegel gibt es zusätzlich das RAL Gütezeichen Software, das ebenfalls nur DIN-konforme Programme ziert. Laut Dr. ▶

Das Merkblatt zur Verwendung der Zeichen ist anzufordern bei: Gütegemeinschaft Software e.V. Postfach 71 08 64, 6000 Frankfurt 71

Schattner, TÜV-Fachmann für Software und Ergonomie sind zur Zeit 45 Programme auf dem Markt, die den Anforderungen gerecht werden.
TÜV und Chip verstehen sich als Interessenvertreter der Anwender. Deshalb fordern beide Institutionen feste Richtlinien für Softwarehersteller. Dadurch und durch die begehrte TÜV-Auszeichnung soll einerseits das Qualitätsbewußtsein der Hersteller-Firmen steigen, andererseits die Kundschaft Vertrauen gewinnen. „Goldgräbermentalität“, wie ein Journalist es nannte, bringt nichts. So sollte es zumindest sein. Kaum glaublich, aber dennoch Tatsache: Über die Hälfte der Software-Geschädigten würden das gleiche Programm wieder kaufen.
(ub)



Proline-Partner 3 Mini Stereo Amplifier ist für Monitore ohne eingebauten Lautsprecher gedacht. In die kleine Box wird entweder ein Kopfhörer oder ein Stereo-Lautsprecher eingesteckt. Der Amplifier kostet ohne Kopfhörer 60 Mark und mit Kopfhörer 70 Mark. Zu beziehen bei GTI, Zimmersmühlenweg 73, 6370 Oberursel/Ts., Tel.: 0 61 71/ 7 30 48



Praktische, ausziehbare Diskettenboxen machen die Softwaresammlung übersichtlich. In eine Plastikbox passen 15 Disketten. Preis: 25 Mark plus 10 Mark Versandkosten (inkl. MwSt). Erhältlich im CompuStore, Fritz-Reuter-Str. 6, 6000 Frankfurt 1.

Neues von Manx

Eine gute Nachricht für alle C-Programmierer. Von Aztec C gibt es jetzt den neuen Quellcode-Debugger 3.6A, eine unentbehrliche Hilfe bei der Fehlersuche in C-Programmen. Mit dem Debugger hat man die Möglichkeit, ein Programm zu testen, ohne es zuvor compilieren zu müssen. Dies bringt erhebliche Zeitersparnis bei der Programmierung. Die Bildschirmmaske ist übersichtlich und funktionell gestaltet und enthält jeweils ein Fenster für Quellcode, Daten und Befehle. Weiterhin besteht die Möglichkeit, Befehlsmakros zu definieren. Alles in allem ist der Source-Debugger die ideale Ergänzung zum Aztec-C-Compiler.

Info: Manx Software, One Industrial Way Eatontown, NJ. 07724, Preis: $ 75,00.



Slow-Scan-Television auf dem Amiga

Amateurfunker aufgepaßt! Mit Hilfe von SSTV ist es jetzt möglich, Grafiken und gescannte Bilder über den Äther zu senden. Nach Angaben des Herstellers soll das Programm eine Bildübermittlung (Senden-Empfangen) in der Qualität eines Schwarzweißfotos gestatten. SSTV ist rein mausgesteuert und sehr komfortabel. Auch das Abspeichern von empfangenen Bildern ist möglich. Zum Lieferumfang gehören ein Interface, das Hauptprogramm sowie viele Hilfsprogramme.

Info: Karl J. Ebensberger, Fällhofstr. 11, 8068 Pfaffenhofen, Tel.: 0 84 41/61 45, Preis: 199 Mark

Kindworks und GoAmiga Text erweitern das Angebot an Textverarbeitungen

Mit GoAmiga Text bringt Softwareland Zürich eine deutsche Version des amerikanischen „Excellent“ auf den Markt. Das Programm ist in der Lage, spaltenweise zu arbeiten und verarbeitet Farbgrafiken. Anhand einer momentan 30 000 Wörter umfassenden Wortbibliothek lassen sich Schreibfehler erkennen. In einer Schulaktion können alle, die sich in der Ausbildung befinden, gegen einen Legitimationsnachweis das Programm zum Sonderpreis von 199 Mark beziehen. Das Angebot ist bis zum 31. 12. 1988 befristet.

Ebenfalls neu ist das Programm Kindworks, das zugleich in sechs Sprachen auf den Markt kommt. Die deutsche Version enthält ein Wörterbuch mit über 150 000 Einträgen zur Rechtschreibkontrolle. Mit geringen Abweichungen läßt sich der Zehner-Tastaturblock wie bei MS-DOS-Programmen zur Cursorsteuerung verwenden (nur in Kombination mit der ALT-Taste). Für das Programm, das ebenfalls Farbgrafiken integrieren kann, wird vom Vertreiber ein Preis von 169 Mark empfohlen.

Info: GoAmiga Text: Softwareland, Franklinstr. 27, CH-8050 Zürich, Tel.: 00 41/1/3 11 59 59

AmigaWelt-Disketten bei Public-Domain-Vertreiber

Wer sich das Abtippen von Listings aus der AmigaWelt ersparen möchte, kann die Disketten zum Heft nun bei einem Vertreiber von PD-Software bestellen. Der Softwareversand M.A.R. versendet Leserdisketten zu den Heften ab 1/88 zum Preis von 60 Schilling pro Diskette zuzüglich 45 Schilling pro Versendung.

Info: M.A.R. Weidengasse 41, A-1100 Wien, Tel.: 00 43/222/62 15 35

C64-Emulation

Für Amiga-User, die ihre C64-Software auf dem Amiga nutzen wollen, hat ReadySoft „The C64 Emulator II“ entwickelt. Laut Hersteller laufen C64-Programme mit dem Emulator um ein zwei- bis fünffaches langsamer als auf dem C 64. Es heißt weiterhin, daß sich der Geschwindigkeitsverlust bei den meisten Programmen nicht bemerkbar mache.

Unser Test ergab dagegen, daß die Hälfte der Software nicht mit Emulator lief. Programme, die mit Emulator funktionierten, waren wegen des starken Geschwindigkeitsverlustes nur eingeschränkt nutzbar. BASIC-Programme, grafikintensive Software und Druckvorgänge, die bereits auf dem C64 sehr langsam waren, verlangsamten sich über den Emulator um ein Weiteres. Besonders problematisch zeigt sich der Emulator bei Spielen. Sprites flackern, wechseln die Farbe oder verschwinden gänzlich. Zudem werden Raster-Interrupts nur unzureichend abgearbeitet.

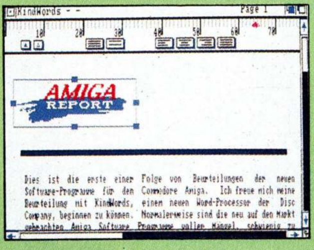
Programme mit Floppyspeedern können mit dem Emulator nicht geladen werden.

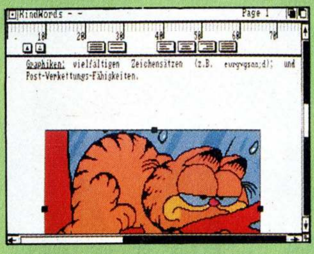
(Th. Meidinger)

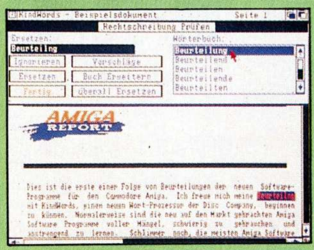
Info: CompuStore, Fritz-Reuter-Str. 6, 6000 Frankfurt 1, Tel. 0 69/56 73 99
Preis: 148 Mark plus 10 Mark Versandkosten

AMIGA™-Textverarbeitung!

KindWords™









GUTE PRESSE?

Die werden sprachlos sein. Wenn Sie die deutsche Version bekommen: KINDWORDS. AMIGA Textverarbeitung!
Natürlich genauso benutzerfreundlich wie die bekannte amerikanische Bestseller-Version enthält KINDWORDS jetzt zusätzlich ein 150.000 Wörter umfassendes deutsches Wörterbuch sowie eine automatische Silben-Trennungsfunktion beim Schreiben.
Außerdem können Sie mit KINDWORDS völlig problemlos Farbgrafiken in Ihre Texte einfügen – für professionelle und eindrucksvolle Dokumente.
KINDWORDS wurde speziell für den Amiga entwickelt – deshalb arbeitet es mit dem Computer – nicht gegen ihn! Das bedeutet: alle Pull-down-Menüs und Hilfsfunktionen werden in vollem Umfang benutzt. Professionelle Features wie Rechtschreibkontrolle und Standardbrief-Verarbeitung sind selbstverständlich enthalten.
Die neue, umfangreiche Lexikon-Funktion und die Möglichkeit der Einfügung von Farbgrafiken machen KINDWORDS zu einem außergewöhnlichen Textprogramm – und das zu einem fast unglaublichen Preis.
Also: Glauben Sie ruhig, was die Presse schreibt!
Wenn Sie mehr über KINDWORDS wissen wollen, schreiben Sie an:
DISC COMPANY EUROPE:
1. rue du Dôme 75116 Paris, France.
Tel: 00 33 1 4553 10 53
Wir schicken Ihnen gerne die ausführliche Broschüre.

mehr KindWords™

DIE PRESSE IST BEGEISTERT!

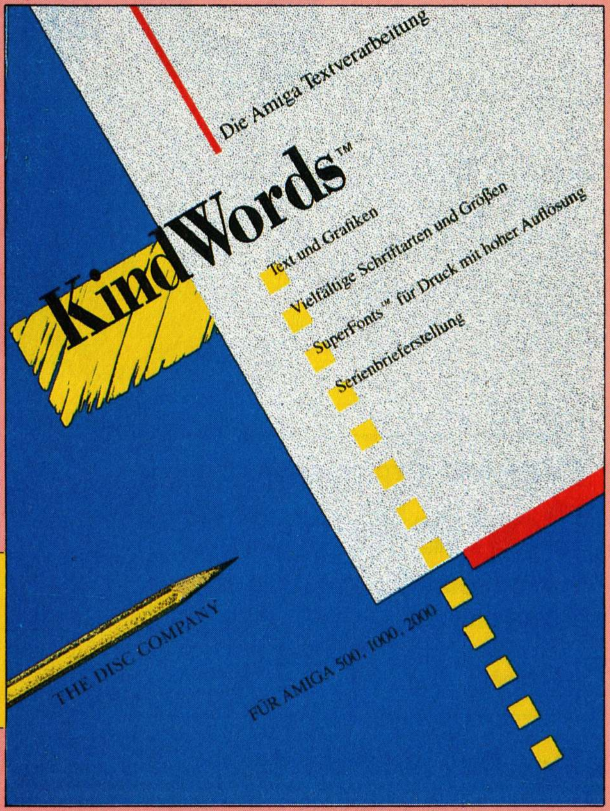
"Das exzellente KINDWORDS enthält alles, was man von einem AMIGA-Textprogramm mit vielen Extras erwarten kann."
ST AMIGA FORMAT

"KINDWORDS ist deutlich billiger als die meisten Amiga-Textprogramme... das Handbuch ist sauber produziert und leicht lesbar ... enthält eine ganze Menge fortschrittliche Features... ein guter Gegenwert für's Geld."
AMIGA COMPUTING

"KINDWORDS ist stilvoll und aufgeräumt...einfach und elegant, mit vielen Vorteilen...KINDWORDS ist ein mit Umsicht geschriebenes Programm, jedes Detail beweist Rücksicht auf den Anwender.
AMIGA USER INTERNATIONAL

"KINDWORDS ist ein starkes und flexibles Programm... ich glaube, daß der logische Aufbau und die starke Konzeption fast alle Anwender begeistern wird!"
COMMODORE MAGAZINE

"Mit KINDWORDS kann man sogar auf billigen DOT-MATRIX-Druckern attraktive Ausdrucke von hoher Qualität produzieren."
AMIGA WORDS



KindWords™
auf 2 Disketten
mit ausführlichem
deutschen Handbuch

DM 169,–
unverbindliche Preisempfehlung

Vertrieb: Schweiz – Elepro AG / Österreich – Karasoft

DAS COMBITEC-GEWINNSPIEL!

Beantworten Sie die drei gestellten Fragen und senden Sie die Lösungen auf einer frankierten Postkarte an Combitec. Einsendeschluß ist der 1. Dezember 1988.

1. Welches Laufwerk wird in der Combitec Disk 3,5 verwendet?
2. Wieviel externe Diskettenlaufwerke sind von dem Amiga 500 ansprechbar?
3. Wie groß ist der serienmäßige Arbeitsspeicher des Amiga 500?

Die eingesandten Karten mit den richtigen Antworten nehmen an der Verlosung teil. Der Rechtsweg ist ausgeschlossen.

1. Preis: 1 Commodore Amiga 500 mit Monitor 1084
2. Preis: 1 Combitec S-RAM 500
3. Preis: 1 Combitec Disk 3,5

Combitec Computer GmbH, Liegnitzer Straße 6 - 6 a, 5810 Witten,
Tel. (0 23 02) 8 80 72

GIB



Combitec Disk 3,5
Diskettenlaufwerk 3 1/2", Busdurchführung für bis zu 3 ext. Laufwerke, Qualitätslaufwerk TEAC Typ: FD 135 FN, Anschluß für Track-Anzeige Typ: Combitec Track-Dis, abschaltbarer Controller, amigafarbenes Gehäuse
VK-Preis: DM 325,–



DEINEM

Combitec Clock 77 S(P)
Normalzeituhr DCF 77, serielle Schnittstelle oder paralelle Schnittstelle (Version P), menügeführte Dateneingabe über Rechner und Datenspeicherung auf Disketten, 4-Schaltausgänge, Anbindungssoftware für Amiga 500, Atari, IBM (Preis auf Anfrage)
VK-Preis: DM 359,–
VK-Preis für Rechneranschlußkabel für o. g. Rechner: DM 24,50



EINE

Weiteres Lieferprogramm:

Combitec 5/3

Steckernetzteil getaktet, 5 Volt, 3 Ampere

VK-Preis: DM 115,–

Amiga 500 VK-Preis DM 940,–
Monitor 1084 VK-Preis DM 568,–

Neu auf dem Markt!
Combitec D-RAM 2/4/8 M für A 500
Externe dynamische Speichererweiterung 2 MB, intern aufrüstbar auf 4 bzw. 8 MB, Einsatz moderner 1 MByte-DRAM-Module, integrierter DRAM-CONTROLLER (0 Wait-states), Gehäuse identisch mit Combitec S-RAM 500.

Einführungsangebot:	
2 MB	DM 1.198,—
4 MB	DM 2.298,—
8 MB	DM 4.498,—



Vom Computertraum zum sendefähigen Film



Schnittplatz mit Schnittsteuereinheit, Bildmischer und Kontrollmonitoren.

In Köln haben sich einige alten Hasen aus den den Bereichen Regie, Bildtechnik, Grafik, Zeichentrick, Systemanalyse und Design zusammengetan, um mit Hilfe des Amiga eine kreative Zusammenarbeit am jungen Medium des Computerfilms zu verwirklichen.



Mit der 1-Zoll-MAZ-Anlage lassen sich die Computersignale in bester Qualität auf Band speichern.



In der Computerecke steht der Bildtrickmischer, bestehend aus AT&T-Grafikstation und Amiga.

Man kennt sie: die metallschimmernden dreidimensionalen Titelzeilen, die taumelnd oder gleitend aus unergründlichen Tiefen auf den Zuschauer zuschweben; die mystischen Kameraflüge aus unmöglichen Perspektiven über gleißende Konstruktionen vor samtfarbenen Himmeln, die sich endlich als Botschaft über den ultimativen Allzweckreiniger entpuppen. Es sind die Eye-Catcher der Werbemedien, neues Augenfutter für den Konsumenten, die zeitgeistgerechten Schlagzeilen des Bildschirmjahrhunderts. Computeranimationen haben noch immer den Reiz des Neuen und die Faszination eines mittels Technik verwirklichten Traums. In der Realität der Medien werden diese technischen Höhenflüge jedoch durch beträchtliche Kosten gebremst. Vierstellige Sekundenpreise für fertige Animationen oder fünfstellige monatliche Leasinggebühren für die zahlenfressende Hardware bedeuten für Auftraggeber und Produzenten ein enormes finanzielles Risiko, da der Einsatz des teuren Geräts bisher nur Werbeagenturen oder finanzstarken Sendern möglich ist. So nützlich und eindrucksvoll eine computeranimierte Senderkennung beispielsweise ist, so wenig schöpft sie die kreativen und innovativen Fähigkeiten des filmemachenden Computers aus. Wer kann sich schließlich schon erlauben, bei solchen Preisen tagelang mit Animationsprogrammen herumzuspielen, um etwas Interessantes, Ungewöhnliches zu realisieren? Und hier beißt sich die Katze in den Schwanz: Welcher tatendurstige Grafiker, Designer oder Videomacher kann auf- und anregende Ideen anbieten oder auch nur verwirklichen, wenn ihn jede Stunde des Probierens so viel kostet wie ein Kurzurlaub?



Bei der Zeichentrick-Animation wird ein waches Auge verlangt.

Kurz gesagt, Computeranimation ist toll, aber unbezahlbar teuer. Oder? Ist es möglich, mit einem populären Computer wie dem Amiga diese Engpässe zu überbrücken? Kann man mit diesem relativ preiswerten Gerät und der in großen Auflagen verkauften Grafik- und Animationssoftware ein professionelles Produkt herstellen, das den Anforderungen der Medien und der Macher gerecht wird oder neue Horizonte für Produzenten und Auftraggeber eröffnet? Kann man mit diesem fantastischen kleinen Computer, der an den Anfängen seiner Karriere einmal ein Spielzeug werden sollte, tatsächlich und realistisch neue Bilder machen, besser informieren oder sich an ungewöhnliche Experimente wagen? Wie fast immer bei solch globalen Fragen ist die Antwort darauf ein klares „Ja, aber . . .“. Ja, man kann professionell mit dem AMIGA arbeiten, wenn man ihn, die Software und die Erfordernisse der Medien sehr gut kennt und gelegentlich über die Geduld eines asiatischen Weisen verfügt — die „Guru-Meditationen“ heißen wirklich zu Recht so. Aber . . ., was heißt eigentlich in diesem Zusammenhang das vielstrapazierte Wort „professionell“? Ständig wird von professioneller Software geredet und geschrieben, vom Amiga als mehr oder weniger professionellem Computer. Nun ist ein Computer genauso professionell wie eine Bratpfanne. Es kommt schließlich auf das Essen an, und wem man es serviert. Menschen, im Technokratendeutsch „brainware“, bestimmen die Ergebnisse der Arbeit mit dem Werkzeug Computer. In diesem Sinn ist der AMIGA professionell, denn er hat alle Voraussetzungen, um damit gute, witzige, unterhaltsame und attraktive Computeranimationen und -grafiken zu machen. Vor allem bietet er eine immens wichtige Voraussetzung zur Teamarbeit: Er ist greifbare, praktikable Realität und nicht eine unbezahlbare Traummaschine. Und nur im Team, mit engagierten Kreativen (und mit Geduld . . .) kann man schließlich „special effects“ in die Welt setzen.

Für die Verwertbarkeit einer Produktion ist das magnetische Aufzeichnungsformat entscheidend. Eine weitere Bearbeitung mit diesem Me- ▶

dium bedeutet immer, daß das Originalmaterial ein- oder mehrmals kopiert wird, wobei die Bildqualität von einer Kopiengeneration zur nächsten abnimmt. Den höchsten Standard bei der Aufzeichnung setzt das 1-Zoll-Videoformat, das bei Sendern und Produzenten weitgehend die Norm bildet. In diesem Format sind alle Bearbeitungsvorgänge wie Mischung oder Schnitt bis in die zehnte Kopiergeneration ohne sichtbaren Qualitätsverlust machbar. Weiter folgen dann Formate wie U-Matic High- und Low-Band, Betacam und in Grenzbereichen Super-VHS oder 8-mm-Video. Diese Formate unterscheiden sich dabei weniger in den Effekt- und Trickmöglichkeiten, da hier die Hardware außerordentlich weit entwickelt ist, als eben in der Wiedergabequalität und den damit gegebenen Bearbeitungsgenerationen. Das für „Low-cost"-Computeranimationen geeignete Aufnahmeformat ist daher keineswegs bei den Heim-Videosystemen zu finden, wenn wir nach einer professionellen Verwertung für den Amiga suchen.

Im Studio werden Animationen Bild für Bild aufgezeichnet

Dazu kommt noch eine weitere Rahmenbedingung: Animationen, ob computergeneriert oder handgezeichnet, bestehen bekanntermaßen aus Einzelbildern, beim PAL-System aus 25 Bildern pro Sekunde. Die verschiedenen Animations-Softwarepakete für den Amiga bieten nun fast alle zwei unterschiedliche Generierungsmöglichkeiten: Die Einzelbilder können als komplette Grafik oder mittels der unterschiedlichen Differenzalgorithmen in gepacktem, verkürztem Format produziert werden, wobei sich gepackte Animationen als zusammenhängende Szenen in Echtzeit direkt aus dem RAM des Rechners abspielen lassen. Das ist sehr eindrucksvoll, und es wäre auch im Sinne von „Low-cost"-Produktion sehr wünschenswert, wenn man einen 30 Sekunden langen, animierten Take auch in 30 Sekunden MAZen, also auf Videoband aufzeichnen könnte. Da eine anspruchsvolle Animation jedoch relativ komplex ist und Kamerafahrten sowie viele bewegte Objekte beinhaltet, treten mit einiger Sicherheit eine ganze Menge Probleme bei der RAM-Animation auf. Eine Wiedergabe aus dem RAM in Echtzeit mit 25 Bildern in der Sekunde ist nicht mehr möglich. Man landet also genau da, wo der konventionelle Trickfilm schon seit Jahrzehnten steht: Animationen bestehen aus Einzelbildern, die einzeln — Stück für Stück — aufgezeichnet und mit erhöhter Bildfrequenz wiedergegeben werden. Dieser Einzelbildschnitt ist bei den eben erwähnten magnetischen Aufzeichnungsverfahren machbar und praktikabel.

Man kann also mit den erhältlichen Softwarepaketen wie Animate3D, Videoscape3D, Apprentice Animator und einigem „Zubehör" wie DPaint, Interchange und Butcher komplexe Animationen erstellen, die berechneten Einzelbildern auf einer Festplatte abspeichern und sie dann im Videostudio über ein gutes Genlock-Interface oder einen Timebase Corrector, der die Studiogeräte und den Computer gemeinsam taktet, aufzeichnen.

Der Einzelbildschnitt wird mit einer Schnittsteuereinheit kontrolliert, die den Aufzeichnungsmaschinen den genauen Schnittpunkt für jedes Bild angibt, den mechanischen Vorlauf der Maschinen synchronisiert und schließlich den bildgenauen Schnitt ausführt. Auch dieser Vorgang kann vom Computer gesteuert werden, wobei für den Amiga eine derartige Steuerungssoftware bereits erhältlich ist.

Genügend Bildschirmpunkte sind die Voraussetzung für echte Professionalität

Die technische Routine bei der Produktion einer Animation sieht so aus: Die Einzelbilder werden nachts gerendert und abgespeichert, am nächsten Tag gesichtet und korrigiert und schließlich wieder in der Nacht rechnergesteuert auf die MAZ geschnitten. Danach stehen — wenn alles geklappt hat — wieder alle elektronischen Tricks des Studios und des Rechners zur Verfügung, um die fertige Animation weiter zu bearbeiten, um hinter die Tricksequenz einen Realfilm zu mischen, weitere Animationen — etwa Titel — zu integrieren oder auch um Musik und Sprache zu unterlegen. Die im Studiobereich geforderte Bildauflösung von 5 bis 7 MHz (etwa 560 Linien) erreicht der Amiga bei Interlace und HiRes leicht, was natürlich beträchtliche Auswirkungen auf die Berechnungszeiten der Einzelbilder hat. Und damit sind wir wieder beim ersten und wichtigsten Glied in der Kette der Komponenten für eine Animationsproduktion: dem Computer. Bis hierher ist die geschilderte Bildbearbeitung eigentlich weniger durch den Rechner als durch die produktionstechnischen Vorgaben gekennzeichnet. Die Kosten für eine Computeranimation sind natürlich zu einem nicht geringen Teil durch die aufwendige professionelle Studiotechnik verursacht, hier versprechen jedoch die zukünftigen Entwicklungen einer durchgängig digitalen Videotechnik eine wesentliche Verbesserung. Auch hier werden sich Consumer- und Profibereich immer stärker annähern, und für eine Verbreiterung der Hardware-Basis sorgen.

Der eigentliche „Knackpunkt" ist natürlich der Amiga, der als Alternative zum spezialisierten, teuren Grafik- und Animationssystem in seiner Leistungsklasse unerreicht ist und es wohl auch noch eine Weile bleiben wird. Auch die Ankündigungen von Commodore und anderen Firmen für die nächste Zukunft weisen in diese Richtung und sind für eine Weiterentwicklung unerläßlich. Das Spektrum für Computeranimation wird sich durch die Verbesserungen bei der Bildqualität und den Rendering-Zeiten noch deutlich erweitern.

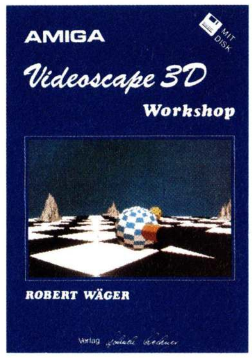
Technisch ist da schon eine Menge geschehen, was vor drei Jahren noch für reichlich utopisch gehalten wurde. Volle 32-Bit-Breite sorgt für erträgliche Rechenzeiten, wobei man bei aller Ungeduld daran denken sollte, daß Bildberechnungen zu den absolut avantgardistischen Anwendungen der Computertechnik zählen. Hochwertige Plattenspeicher mit 100 MB und mehr sind erhältlich und auch dringend erforderlich,

Fortsetzung auf Seite 22

Computerbücher und Software aus dem Verlag *Gabriele Lechner*

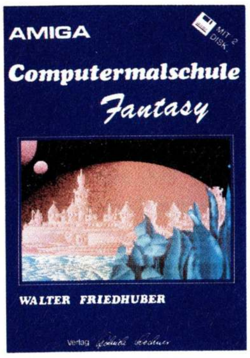
zuverlässig – aktuell – informativ

Die Bücher mit der persönlichen Note



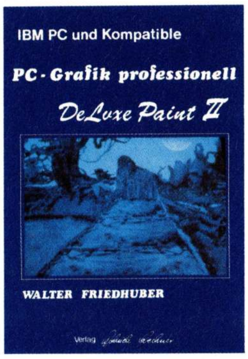
Die WORKSHOP-REIHE
Videoscape 3D Workshop
Autor: Rober Wäger
250 S., 30 Abb., kt. inkl. 1 Disk.
ISBN 3-926858-03-6 DM 59,00
Sculpt 3D Workshop
Autor: Harald Schott
250 S., ca. 30 Abb., kt.,
inkl. 1 Disk.
ISBN 3-926858-10-9 DM 59,00
erscheint Dez. 1988
Silver Workshop
Autor: Gabriele Lechner
250 S., ca. 40 Abb., kt.,
inkl. 1 Disk.
ISBN 3-926858-12-5 DM 59,00
erscheint Dez. 1988

Computermalschule Fantasy

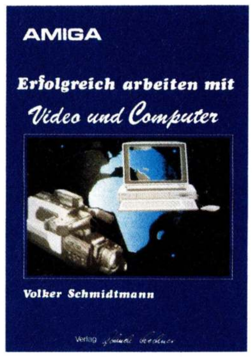


Autor: Walter Friedhuber
204 S., ca. 40 Abb., geb.,
inkl. 2 Disk.
ISBN 3-926858-06-0 DM 59,00
Computermalschule Landschaften
Autor: Gabriele Lechner
ca. 170 S., ca. 50 Abb., geb.,
inkl. 2 Disk.
ISBN 3-926858-05-2 DM 59,00
Computermalschule Trickfilm-zeichnen
Autor: Walter Friedhuber
185 S., ca. 50 Abb., geb.,
inkl. 2 Disk.
ISBN 3-926858-07-9 DM 59,00

Für IBM PC und Kompatible
PC-Grafik professionell:
DeLuxe Paint II
Autor: Walter Friedhuber
500 S., ca. 200 Abb., kt.,
inkl. 1 Disk.
ISBN 3-926858-16-8 DM 79,00
Das Erfolgsbuch nun auch für den IBM-PC. Randvoll mit Tips und Tricks.

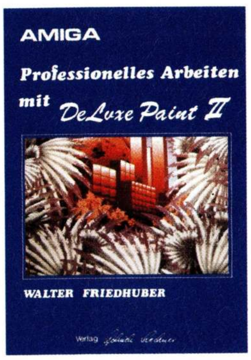


Faszination Video und Computer



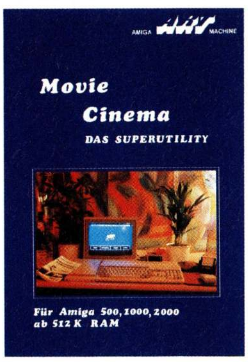
Erfolgreich Arbeiten mit Video und Computer
Autor: Volker Schmidtmann
ca. 320 S., ca. 30 Abb., kt.
ISBN 3-926858-14-1, DM 69,00
erscheint Nov. 1988

Der Bestseller



2. neu überarbeitete Auflage
Professionelles Arbeiten mit DeLuxe Paint II
Autor: Walter Friedhuber
540 S., ca. 200 Abb., kt.
ISBN 3-926858-04-4 DM 69,00
Das Standardwerk zum Mal- und Zeichenprogramm
DeLuxe Paint II
Disk-Set zum Buch unverb. VK DM 39,00 (inkl. 2 Disk)

Die Supersoftware zu DeLuxe Paint II



MOVIE CINEMA
ist ein leistungsstarkes Utility für alle DeLuxe Paint-Benutzer, die Interesse am Trickfilmzeichnen haben. Bestell Nr. 7100
unverb. VK DM 69,00

Aktuelle Software

Videoeffects 3D	unverb. VK 495,00 DM
Aegis Videoscape 3D PAL 2,0	unverb. VK 385,00 DM
Aegis Video Titler 1.1	unverb. VK 269,00 DM
Aegis Lights Camera Action	unverb. VK 169,00 DM
Fantavision	unverb. VK 169,00 DM
Fancy 3D-Fonts (für Sculpt und Animate)	unverb. VK 148,00 DM
DeLuxe Productions	unverb. VK 489,00 DM
Synthia (Top Musikprogramm)	unverb. VK 228,00 DM
NEU: Im Brennpunkt: THE DIRECTOR (Daten, Fakten, Tips zum Director, Buch ca. 120 S.)	unverb. VK 29,80 DM

Sonderpaket:
Movie Cinema
+ Professionelles Arbeiten mit DeLuxe Paint II
+ Disk 1 und 2 zum Buch Gesamtpreis unverb. VK 149,00 DM

Video und Computer

Digi View 3,0	unverb. VK 395,00 DM
RGB-Splitter VCP 7000	unverb. VK 385,00 DM

Neue Version!
Electronic Design PAL Genlock (für Amiga 500, 1000, 2000A, 2000B) unverb. VK 555,00 DM

Hotline für Videoberatung von 14 bis 17 Uhr Tel. 089/834 05 91

Softwarelandprodukte:

Go Amiga Titel	unverb. VK 89,00 DM
Go Amiga Datei	unverb. VK 199,00 DM
Go Amiga Text	unverb. VK 298,00 DM

Sonderaktion:
Go Amiga Text für Schüler, Studenten und Auszubildende jetzt zum Preis von unverb. VK 199,00 DM

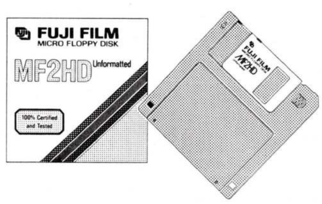
Auslieferung:
BRD
Verlag
Gabriele Lechner
Planegger Str. 1
D-8000 München 60
Tel. 0 89-834 05 91

Schweiz
Softwareland AG
Franklinstr. 27
CH-8050 Zürich
Tel. 01-311 59 59

Österreich
Fa. Intercomp
Heldendankstr. 24
A-6900 Bregenz
Tel. 0 55 74-273 44

Filiale:
Y. Schott, Kloberstraße 6, 6503 Mainz/Kastel, Tel. 0 61 34 / 67 86
Enterprise Company Station, Achterstraße 19, 3110 Uelzen 1, Tel. 05 81/152 40

FUJI FILM DISKETTEN



Kinnhaken mit Dynamic Drums

Daß ein Stuntteam im Elefantengehege des Münchner Tierparks von der Feuerwehrleiter in Pappkartons springt und wilde Schlägereien vorführt, mag ja ungewöhnlich sein. Aber daß zwischen Dickhäutern und Kartonhaufen ein Amiga eingesetzt wird, das ist schlichtweg sensationell.



Mit voller Wucht den Fuß dem Gegner ins Gesicht gerammt — das muß weh tun.



In Wahrheit simuliert ein digitalisierter Sound den Aufprall der Ferse. Die Einsatzmöglichkeiten des Amiga scheinen unerschöpflich.

Besucher des Münchner Tierparks drängeln am Zaun des Elefantengeheges. Die grauen Rüsseltiere werden kaum beachtet, da sich direkt daneben zwei junge Männer einen aufregenden Faustkampf liefern. Sie verpassen einander gehörige Kinnhaken und Magenschwinger — so scheint es zumindest, denn sie holen kräftig aus, stürzen zu Boden und krümmen sich vor Schmerz. Allerdings ist kein Laut zu hören. Natürlich nicht, denn dies ist eine gestellte Schlägerei. Die beiden Stuntmen Sören Schaper und Michael Rausch haben lange darauf trainiert.

Ihr Kollege Guido Hanke am Lautsprecherpult wird jetzt der Darbietung mit Schlaggeräuschen eine herbere Note geben. Jedesmal, wenn ein Kollege zugeschlagen hat, drückt er eine Taste am Computer. Bisher ist kaum aufgefallen, daß neben Musikanlage, Reglerkasten und Fernsehmonitor ein Amiga 500 steht. Auf dem Bildschirm ist das Menü des Musikprogrammes „Dynamic Drums" zu erkennen. Anstatt die Tasten mit musikalischen Fragmenten zu belegen, kamen die Bavaria Stuntleute auf die Idee, Schlaggeräusche zu digitalisieren und im richtigen Moment mit einem Tastendruck auszulösen.

Das ist aber beileibe nicht alles, was das Bavaria Stunt Team zu bieten hat. Zu ihrem Repertoire gehört fast jede gestellte Gefahrensituation. Sprünge, Unfälle, halsbrecherische Klettertouren und Verfolgungsjagden beherrschen sie aus dem FF und demonstrierten dies bereits in „Tatort", „Derrik" und „Der Alte". Im Münchner Zoo führen sie zwei Spezialitäten vor. Michael Rausch, eingepackt in zwei Schichten Schutzkleidung, wird mit Spezialbenzin übergossen und angesteckt. Meterhohe Flammen und Temperaturen bis zu 800 Grad entstehen. Sören Schaper löscht die „lebende Fackel" mit einem Feuerwehrschlauch. Unversehrt wikkelt sich der nasse Stuntman aus dem Anzug.

Vor dem Elefantenhaus ist ein weiterer Stunt vorbereitet worden. Ein Feuerwehrauto fährt rückwärts neben einen riesigen Quader aus Pappkartons. Die Leiter wird bis zur luftigen Höhe von 15 Metern ausgefahren. Sören Schaper klettert hinauf, springt und verschwindet mit lautem Gepolter im Papphaufen. Und wieder ist niemand zu Schaden gekommen.

Wie kommt man als Stuntman auf die Idee, den Amiga mit in's harte Geschäft zu nehmen? Sören Schaper erzählt, daß man sich ja schließlich neben dem Job auch mit dem Amiga beschäftigt und am Kingsoft-Spiel „Emetic Skimmer" mitgearbeitet hat. Warum also nicht den Amiga zur Vervollkommnung eines Stunts einsetzen?

(ub)

CIT

Computer und Innovative Technologien

Ihr Partner für den professionellen Einsatz des Commodore Amiga in Forschung, Technik und Industrie

AMIGA ALS CAD - ARBEITSPLATZ

X-CAD ist das professionelle 2D-CAD-Programm für Zeichner, Designer und Ingenieure für den Commodore Amiga, das die Leistungsfähigkeit dieses Rechners voll nutzt. Der Anwender hat viele Möglichkeiten der Ein- und Ausgabe, um produktiv Zeichnungen zu erstellen. Mit verschiedenen Zusatzmodulen ist X-CAD ein flexibles Instrument für diverse Fachrichtungen (Elektronik, Maschinenbau, Architektur).
Kurzübersicht X-CAD Designer
+ Grafikelemente : Punkt, Linie, Polygon, Ellipse, Bogen, Schraffur, Text (div. Fonts)
+ Konstruktionshilfen : Parallele, Senkrechte, Tangente, etc.
+ Halbautomatische Bemaßung
+ Layertechnik mit 256 Layern und 8 Depth (Tiefen)
+ Definition von eigenen Symbol-Bibliotheken
+ Eingabe mit Tastatur, Mouse, Tablett
+ Ausgabe auf Penplotter, Matrix-, Laser-, Thermotransfer-, Farbelektrostatikdrucker
+ Erstellung eigener Menüs für Bildschirm und Tablett
+ Automenüs mit Natural-Language-Interface
(z.B. DRAW LINE TANGENTIAL :Entity :Entity)

X-CAD Designer V1.3 **Engl. Handbuch**	**995,- DM**
X-CAD Designer V1.3 **deutsches Handb.,**	**1.495,- DM**
X-CAD TURBO-Designer **deutsches Handb.,**	**1.995,- DM**

Weitere Produkte als Erweiterungen für X-CAD Designer :

AutoCAD-Converter Umwandlung von AutoCAD DXF-Files ins X-CAD-Format	**450,- DM**
X-MACRO C-ähnlicher Interpreter zur Varianten-Konstruktion im Maschinen- oder Anlagenbau	**2.995,- DM**
X-LIB Hydraulik/Pneumatik	**495,- DM**
X-LIB Verfahrensfließbilder	**995,- DM**
X-LIB Rohrleitungsbau	**995,-DM**
X-LIB Fließbilder EDV	**395,- DM**

CAD-PAK/1
Minimal-KOnfiguartion zum vernünftigen Arbeiten mit X-CAD Designer
- Amiga 2000 (1MB + 3,5" LW)
- A2090 Controller mit 20 MB Festplatte
- Noninterlaced Color-Adapter PAL- Version
- EIZO Flexscan 8060S 14" Monitor
- X-CAD Designer V1.3 Deutsch

Weitere Komplett-Konfigurationen auf Anfrage.

AMIGA ALS WORKSTATION

Wir erweitern Ihren Amiga 500 oder 2000 mit 32-Bit-Prozessoren von Motorola und Inmos. Damit haben Sie die Leistung einer 32-Bit-Workstation mit den grafischen Leistungen des Amiga auf Ihrem Schreibtisch.
32-Bit-Amiga mit Motorola 68020/030/881/882
Mit den TURBO-Karten von CSA können Sie ihre alte Software laufen lassen. Die bestehende Software läuft mit 32-Bit-Speicher zwischen 200-500%, mit Matheprozessor sogar 1000-2000% schneller. Einige Software unterstützen den Matheprozessor z.B. Sculpt/Animate 3D, X-CAD TURBO Designer.

TURBO2000-CPU/020/881A	68020/14MHz 68881/14MHz	**2995,-DM**
TURBO2000-CPU/020/882B	68020/14MHz 68882/20MHz	**4195,-DM**

NEUHEIT
TURBO-PACK/1 — **6995,- DM**
+TURBO2000-CPU/020/881A
+Dragestrip
- (Kickstart 1.22 32-Bit) mit Autoboot
- 4MB 32Bit-Speicher (erweiterbar auf 16MB)

TDP2000/A — **5495,- DM**
- Megalink01 mit 1x T800 mit 1MB Speicher (erweiterbar auf 4x T800 mit 1MB)
- Amiga-Link2000 mit 5-20 MBaud

AMIGA MIT HOHER AUFLÖSUNG

Microway Noninterlaced Color-Adapter — **1.395,- DM**
Diese Grafikkarte wird im Videoport des Amiga 2000 (Type B) betrieben und ermöglicht die augenschonenede Benutzung von Software im Interlaced-Modus. Die maximale Auflösung beträgt 702x512 Pixel. Sie hat keinen Einfluß auf Software und muß mit einem Multisync-Monitor betrieben werden (31,5kHz/50Hz).

Megavision02 — **3.695,- DM**
Grafikkarte mit Grafikprozessor HD63484 und 1MB DRAM Bildspeicher. Anschlußfertig an Amiga 500/1000/2000. Auflösungen von 512x512 in 16000 Farben bis 800x600 in 256 aus 256000 Farben Noninterlaced. Anschluß an Multisync-Monitore oder Spezialkabel A1084.

Komplett-Pakete
Microway Noninterlaced Color-Adapter mit :

Eizo Flexscan 9070S 16"	**3.795,- DM**
EIZO Flexscan 8060S 16"	**3.295,- DM**
NEC Multisync II 14"	**3.375,- DM**
NEC Multisync XL 20"	**8.325,- DM**
Eizo Flexscan 8060S 14"	**3.295,- DM**

AMIGA IM NETZWERK

Wir verbinden Ihren Commodore Amiga 2000 und 500 mit den Ethernet-Karten von Ameristar Technology Inc. mit Systemen von anderen Herstellern (z.B. SUN™, DEC, VAX™, Apollo Domain™) zu einem leistungsfähigen Verbundsystem zur Steigerung Ihrer Produktivität. Sie sparen Zeit und Geld bei gemeinsamer Benutzung von Hardware (z.B. Massenspeicher, Drucker und Plotter) und Software (z.B. Datenbank, Libraries). Die Netzwerk-Software ist TCP/IP und NFS-Implementation (Network-Files-System von SUN). Mit den Zusatz-Packeten für X-Windows oder NeWs kann der Amiga als Grafikterminal für SUN oder andere Rechner dienen und vom Amiga dessen Software benutzen.

NFS-Client-Packet
TCP/IP- Hanlerund NFS-Software (FTP, TELNET, RLOGIN usw.)

mit AE2000 für Amiga 2000	**2650,-DM**
mit AE500 für Amiga 500 (Erweiterbar mit 2MB DRAM)	auf Anfrage
X-Windows- oder NeWs-Emulation	auf Anfrage

AMIGA ALS SPEICHERGIGANT

GIT-Subsysteme für spezielle Anwendungen
Bei Platzproblemen im Amiga liefern wir Subsysteme für Massenspeicher oder Coprozessoren (Transputer oder Grafikkarten). Folhgende Gehäuse sind lieferbar.

Technische Daten	GIT2000	GIT 2500
Kapazität	5x5,25"HH	5x5.25" + Megavision02
Netzteil	200W	200W
Preis	895,- DM	1250,- DM

GIT2000 + A2090-Controller + SCSI-HD 65MB 40ms	**3.995,- DM**
GIT2000 + A2090-Controller + SCSI-HD 107DM 28ms	**5.495.- DM**
GIT2000 + A2090-Controller + SCSI-HD 150MB 28ms	**6.325.- DM**

Andere Konfigurationen auf Anfrage
(Backup-Tapes für A2090 in Kürze lieferbar)

Sonderangebot für alle Amiga 2000-Besitzer

Commodore Controller A2090 + Seagate ST251-1(28ms)	**1.850,- DM**
Nachrüstkit mit Seagate 251-1(28ms)für A2090-Controller	**995,- DM**

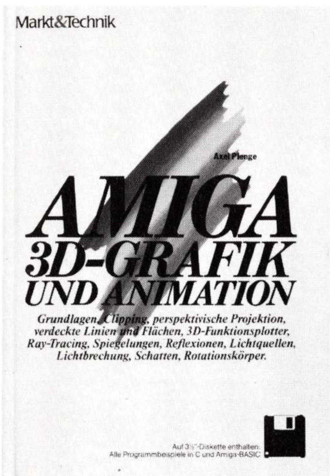
Alle Preise verstehen sich zuzüglich Fracht.
Die Lieferung erfolgt per UPS mit Nachnahme.
Es gelten unsere allgemeinen Verkaufs- und Lieferbedingungen.

CIT – Wiese

Maassenstr. 10 ✱ 4235 Schermbeck ✱ Tel. 0 28 53 / 40 99

Dreidimensionales und Bewegtes

Wer sich bisher keine komplizierte Grafiksoftware zugetraut hat, bekommt Hilfe mit dem Buch „Amiga 3-D-Grafik und Animation". Es vermittelt die entsprechenden mathematischen und programmiertechnischen



Grundlagen. Begonnen wird mit dem essentiellen Basiswissen für Farbgrafik, insbesondere in bezug auf die Grafikansteuerung in BASIC und C. Im darauf folgenden Kapitel wird mit einfachen 2-D-Operationen ein Fundament für 3-D-Programmierung gelegt; dazu gehören Rechenoperationen vom Kaliber der Matritzenmultiplikationen. Anschließend werden geometrische Grundformen wie Kreise und Ellipsen besprochen und die Anwendung von Spiegelung und Rotation erklärt. Im Anhang stehen die dazugehörigen mathematischen Funktionen wie Sinus und Cosinus. Dann geht es in die dritte Dimension und in die Bewegung. Transformationen im Raum werden anhand verschiedener Projektionsarten veranschaulicht. Wo Körper in Bewegung sind, überlappen sie sich mit räumlich dahinterliegenden Linien und Flächen. Im Buch werden zu diesem Problem Lösungswege vorgestellt.
Anhand des bis dahin Gelernten ließen sich bereits ganz ansehnliche Effekte erzielen. Doch erst mit dem Ray-Tracing-Verfahren werden die Ergebnisse verblüffend. Lichtreflexe, Schatten und Beleuchtung räumlicher Objekte sind so möglich und variabel. Spezielle Erwähnung finden abschließend die Rotationskörper, mit denen sich interessante Licht- und Schattenspiele vollführen lassen. Um die Sache zu komplettieren, werden alle verwendeten Library-Routinen im Detail erklärt.
In didaktisch sinnvollen Schritten wird der Lernbegierige in die Materie eingewiesen, wobei Programmbeispiele auf mitgelieferter Diskette das Verständnis erleichtern. Im Literaturverzeichnis wird auf weiterführende Werke zum Thema verwiesen.
(S. Quinkertz)

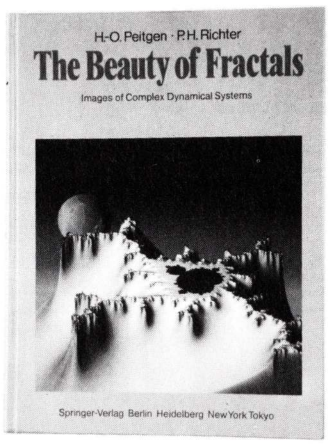
Amiga 3D-Grafik und Animation, Axel Plenge, Markt&Technik Verlag, Haar bei München 1988, 376 Seiten, 1 Diskette, 69 Mark, ISBN 3-89090-526-9

Ästhetisches Chaos

Unmengen von Apfelmännchen-Routinen stürzen Amigas und andere Computer in ungeahnte Rechentiefen. Doch nicht nur die faszinierenden Apfelmänner zeugen von den Gesetzen eines mathematischen Chaos'. Weitere Resultate der „jungen" Chaosforschung sind beispielsweise die Julia-Menge, als Untermenge der Mandelbrot-Menge, und „seltsame Attraktoren" für chaotische Bewegungen („Dreikörperproblem").
Computerlaien wie auch Erforscher mathematischer Gesetzmäßigkeiten sind gefesselt von geheimnisvoller Ornamentik und oszillierender Farbenpracht, die entsteht, während die Konvergenzkriterien unendlicher Reihen der komplexen Zahlenebene analysiert werden. Von solchen fraktalen Unglaublichkeiten handelt „The Beauty of Fractals".

Verfasser sind zwei deutsche Professoren, die in der Grundlagenforschung der Experimentalmathematik arbeiten; sie haben sich im Speziellen den Fraktalen verschrieben. Mit seinem Buch spricht das Professorenduo den Ästheten und den Mathematiker an. Wer sich nicht nur am Bilderbuchcharakter erfreuen, sondern auch die mathematischen Grundlagen begreifen will, kann seine Neugier stillen. Dabei muß er sich mit der Herleitung teilweise recht haariger Differenzialgleichungen konfrontieren.

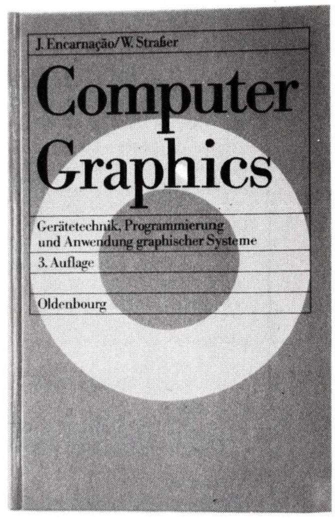
Doch wird der Leser behutsam in's Chaos geführt. Zuerst erfährt er, wie die Mandelbrot-Menge zu ihrem kulinarisch klingenden Namen kam. Erst dann werden ihm häppchenweise die mathematischen Voraussetzungen verabreicht. Trotzdem spielt sich die Auseinandersetzung mit diesem neuen Wissenschaftszweig auf hohem Niveau ab, wobei eingestreute Bilder die Lektüre auf angenehme Weise unterbrechen.
(W. Häring)



The Beauty of Fractals (engl.), H.-O. Peitgen, P.-H. Richter, QA 447.P45 1986 516 86-39.17, Springer-Verlag Berlin Heidelberg 1986, ISBN 3-387-15851-0

Grafik im System

„Computer Graphics: Gerätetechnik, Programmierung und Anwendung" stammt aus der Feder der Professoren Encarnacao und Straßer. Gegenstand der Abhandlung ist das Prinzip des „Grafischen KernSystems" (GKS). GKS ist ein interaktiv mit seinem Anwender kommunizierendes Grafiksystem, das sich besonders für computerunterstütztes Konstruieren an Workstations eignet. Deren Konzeption weist Verwandschaft zu einer Amiga-Grafiksoftware wie Intuition auf.



Probleme, die bei der Programmierung derartiger grafischer Systeme auftauchen, werden hier zerpflückt und unter die Lupe genommen. Am Beispiel des GKS werden die Anforderungen an Grafiksysteme erläutert, die bei Datenverarbeitung, Hardwaretechnik und Gra-

fikprogrammierung gestellt werden. Ferner wird das Problem gewälzt: Wie programmiert man grafische Dialoge so, daß sie ihrem Benutzer eine „natürliche“ Kommunikationsmöglichkeit bieten? Des weiteren werden Verfahren der dreidimensionalen Darstellung von Daten vermittelt. Dabei wird der methodische Bereich von Koordinatentransformation bis Ray Tracing nicht nur gestreift, sondern umfassend behandelt. Abschließend vertiefen die Autoren Datenkonzepte zum Entwurf komplexer Grafiksoftware, ebenfalls die Problematiken bei CAD/CAM. Trotz weitgefächerter Thematik ist jeder Bereich gründlich ausgearbeitet. Die Anforderungen an den Leser sind hoch; manche Kapitel verlangen mathematisches Grundwissen. Dennoch bietet das Buch eine Heranführung an das heiße Eisen Computergrafik. Es eignet sich damit als ergänzende Lektüre für Fachstudenten als auch für den Autodidakten.

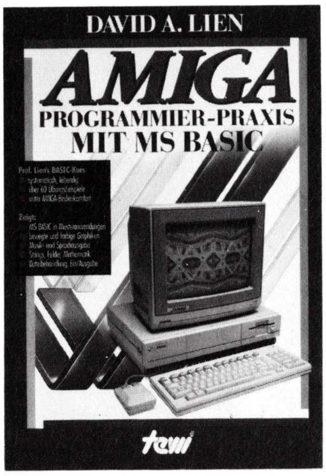
(W. Häring)

Computer Graphics, José Encarnaçào, K. Straßer, Oldenbourg Verlag, München, 452 Seiten, ca. 50 Mark, ISBN 3-486-20640-0

BASIC fundamental

Zur Standardausstattung des Amiga gehört ein BASIC-Interpreter, dem ein recht umfangreiches Handbuch beiliegt. Allerdings werden Einsteiger ohne Vorkenntnisse ihre Not haben, anhand dessen die BASIC-Programmierung zu erlernen, da das Manual wie ein Referenzwerk aufgebaut ist. In diese Bresche springt David Liens Buch „Amiga — Programmierpraxis mit MS BASIC“, mit dessen Hilfe die Grundbegriffe von Microsoft-BASIC auf dem Amiga auf angenehme und doch gründliche Art und Weise vermittelt werden. An keiner Stelle fällt unangenehm auf, daß die amerikanische Urversion bereits drei Jahre alt ist.

Nach einer speziell auf den Amiga abgestimmten Einleitung, die sich mit der Bedie-



nung des Interpreters und Eingabe von Programmen befaßt, wird der allgemeine BASIC-Befehlssatz erklärt. Daran schließen sich Kapitel über Grafik an, über Sound, Sprache, Dateibehandlung sowie Ein- und Ausgabe. Jedes Themengebiet wird durch Übungsaufgaben mit Musterlösungen vertieft. Auf das Problem Fehlersuche – Debugging – geht der Autor gesondert ein. Erklärungen zu allen Fehlermeldungen und ein ausführliches Stichwortverzeichnis finden sich im Anhang.

Bei aller Fachlichkeit ist das Lesen dennoch vergnüglich. Randbemerkungen und kleine Witzeleien lockern das Sujet auf. Logikoperatoren werden anschaulich durch Gatter erklärt. Nicht neu? Doch! Der Autor meint nämlich Gatter auf Kuhweiden. Trotzdem bleibt er nicht an der Oberfläche der Thematik, sondern vertieft Problemfälle und behandelt Fragen, die sich jedem Einsteiger stellen.

Bei der Übersetzung des Buches ließ man es nicht an Sorgfalt fehlen, was gerade in der EDV-Branche eine Ausnahme ist. So blieb der Sprachstil erhalten, Beispielprogramme und Übungen sind übersetzt und viele sprach- und landesabhängige Umstände an deutsche Verhältnisse angepaßt worden.

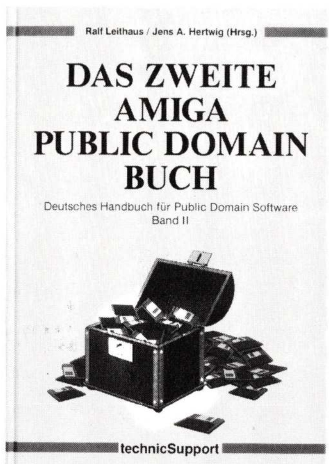
Wenngleich dem Fortgeschrittenen womöglich nicht genügend Amiga-spezifisches geboten wird, so ist dies Buch doch dem BASIC-Einsteiger uneingeschränkt zu empfehlen.

(R. Babel)

Amiga Programmierpraxis mit MS BASIC, David A. Lien; Softcover, te-wi Verlag, München, 1986, 404 Seiten, 59 Mark, ISBN 3-921803-69-1.

Auf PD-Schatzsuche

Mit dem „Zweiten Amiga Public Domain Buch“ reagiert technicSupport auf die enorme Nachfrage nach dem ersten Band. Der Erfolg war in der Tatsache begründet, daß es bis zur Herausgabe des ersten Bandes keine deutschsprachigen PD-Anleitungen gegeben hatte. Band II bringt die User jetzt auf den neuesten Stand. Außer der bekannten Fish-, Panorama- und Faug-Disks wird kanadische RPD-Software vorgestellt. Die RPD-Reihe weicht vom Konzept der Fishdisks ab und bietet den deutschen PD-Freunden Neues und Nützliches. Die Aufmerksamkeit der Autoren gilt aber nicht nur Usern, die sich prinzipiell auskennen und lediglich an Novitäten interessiert sind. Einsteiger werden im Schnellverfahren in den Umgang mit Workbench, CLI und Verzeichnissen eingewiesen, um die Anleitungen problemlos in die Praxis umsetzen zu können. Auf 240 Seiten stehen die Anleitungen zu 46 ausgewählten Programmen. Die Beispiele decken die Bereiche Anwendungen, Grafik, Animation und Musik, Arbeits- und Programmierutilities ab. Darunter findet sich so Ungewöhnliches wie ein Persönlichkeitstest à la Freud, ein mausunterstütztes Ray-Tracing-Programm, eine Spezial-Workbench und „das Ende des Gurus“. Aufruf, Zweck, Beschreibung, Installation und Bedienung der Programme sind ausführlich dokumentiert. Am Ende jeder Beschreibung wird auf die Eigenarten der betreffenden Software hingewiesen. Bei dem jeweiligen Programm-



autoren — Adresse ist angefügt — kann man sich Informationen und Programm-Updates besorgen. *(ub)*

Das zweite Amiga Public Domain Buch, R. Leithaus, Jens A. Hertwig; technic-Support, Berlin (West) 1988; 384 Seiten, 49 Mark, ISBN 3-926847-05-0.

Fortsetzung von Seite 16

da für eine mehrminütige Animation oft 40—50 MB Speicherplatz anfallen, wenn die Einzelbilder vor dem Schnitt noch nachbearbeitet werden sollen.
Die Bildqualität ist im Interlace-Betrieb für den Großteil der produzierten Animationen und Grafiken befriedigend, und für das gestreßte Auge des Benutzers kann ein guter Monitor sorgen. Diese Aufrüstungen kosten natürlich einiges, aber wenn der Amiga als Keimzelle eines Animationssystems derart bestückt ist, kann er immer noch mit mindestens doppelt so teuren Geräten gleichziehen. Die ersten Computerfilm-Anwendungen — wie sie heute schon von wenigen, ausgefuchsten Spezialisten auf dem Amiga produziert werden — sind nur der Anfang. Weitere Möglichkeiten für preiswerte und gute Computerbilder werden eigentlich nur von der kreativen Phantasie und dem innovativen Mut der Künstler und der Techniker eingeschränkt.

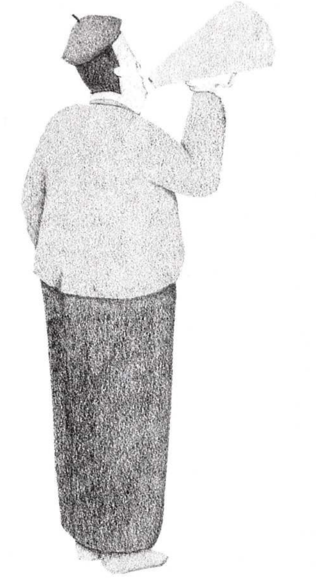
Computeranimationen lassen sich schneller realisieren als am Zeichentisch

Die „seltsamen Vögel" unter den Film- und Videomachern, wie Floyd Wray die Produzenten von Animationen im Animate3D-Handbuch liebevoll nennt, sitzen heute allerdings noch tage- und wochenlang an einem 30-Sekunden-Spot. Nehmen wir zum Beispiel die Produktion des Titels für einen Musikclip: Es sollen — passend zum Text und zum Rythmus des Stückes — animierte Buchstaben durch eine fraktale Landschaft tanzen. Die Buchstaben werden über das Zeichentablett in Sculpt3D eingelesen und erhalten mit der Extrude-Funktion die räumliche Tiefe. Mit Interchange werden dann die Sculpt-Dateien in das Videoscape-Format umgewandelt und dort schließlich animiert. Die genaue Choreographie ist im Storyboard — einem gezeichneten Drehbuch mit Hauptphasen und detaillierten Bildausschnitten — festgelegt und mit dem Auftraggeber abgesprochen. Videoscape kommt zwar bekanntlich in der aktuellen Version noch sehr wenig dem Benutzeroberflächen-verwöhnten Anwender entgegen, bietet dafür aber — trotz aller Einschränkungen in der Farbwahl — eine schnelle und flüssige Animation, die für die Buchstaben in unserem Titel durchaus geeignet ist. Für detaillierte Konstruktionen eignet sich das Animationsprogramm Animate3D auf der Basis des Sculpt3D, wobei die Renderingzeiten mit dem Turboboard kurz gehalten werden. Animate3D ist allerdings ein wirklich ausgefeiltes und zukunftsträchtiges Ray-Tracing-Animationssystem, das weder in der Logik noch im Bedienungskomfort der Software der „Grafik-Jumbos" nachsteht. Die fraktale Landschaft für unser Videoclip kommt schließlich — klein aber fein — aus einem Public-Domain-Programm. Da dieses Programm Dateien erzeugt, die weder mit Interchange noch mit anderen Konvertern in einen der Animationsstandards umgewandelt werden können, wird dieser Hintergrund einzeln produziert und im Studio per Mischer in die zweite Bildebene gelegt. Die fertige Animation wird dann geschnitten und auf den gemischten Soundtrack bildgenau angelegt. Hier zeigt sich die Stärke einer vollwertigen Videotechnik bei Computeranimationen: Die Sequenzen können auf dem am besten geeigneten Programm generiert werden, ohne Rücksicht auf Kompatibilität der Daten einfach einzeln aufgezeichnet und dann per Video zum Endbild gemischt werden. Andererseits spielt hier wieder der Amiga seine Stärken aus, denn auch auf einem großen Rechner läßt sich eine derartige Vielfalt an Programmen einsetzen; diese Flexibilität treibt aber die Produktionskosten in die Höhe.
„Das künstliche Bild" — so wurden vor etwa 150 Jahren die ersten Fotografien genannt. Die künstlichen Bilder des 20. Jahrhunderts liefern die Computer. Die Künstler, die Grafiker, Zeichner und Filmemacher, werden auch dieses, so abstrakt scheinende Medium weiter für sich entdecken.

Kreativer Mittelpunkt bleibt der Mensch

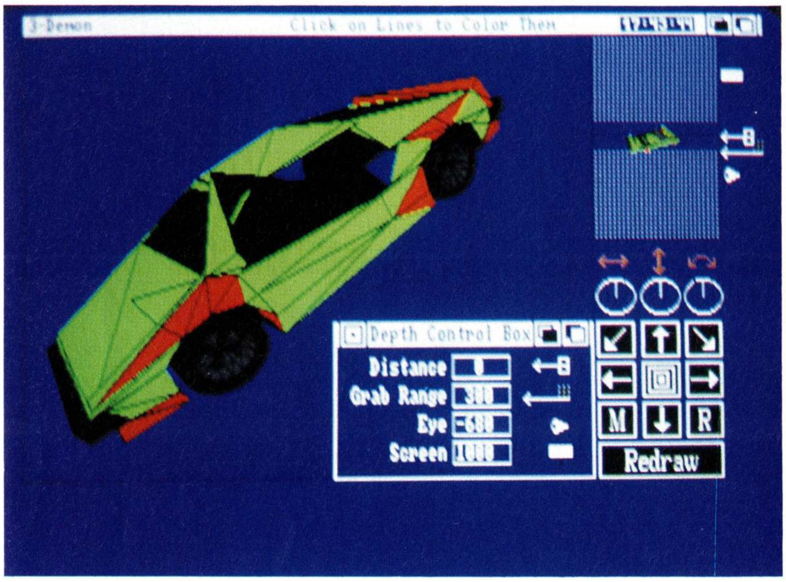
Bereits jetzt zeichnet sich eines ganz klar ab: die Hardware ist und bleibt ein Instrument in den Händen der Profis. Ohne Kreativität, Neugierde und Lust an der Innovation bleibt jede noch so teure Workstation ein Haufen Silicon und Blech. Der Amiga als kleines, flexibles Werkzeug erleichtert eindeutig die interdisziplinäre Zusammenarbeit der Spezialisten am neuen Medium Computerfilm. Niemand redet der künstlich erschaffenen Medienwelt, mit Schauspielern aus dem Rechner und Landschaften aus Bits und Bytes, das Wort. In den „non-picture"-Bereichen, in denen Realbilder nicht existieren oder den Ansprüchen nicht genügen, können und werden Computeranimationen ihren Stellenwert behalten und ausbauen. Die Auswirkungen auf künstlerischem Gebiet sind überhaupt noch nicht absehbar. „Ein Bild sagt mehr als tausend Worte": Das Sprichwort gilt noch immer, erst recht dort, wo Worte und natürliche Bilder fehlen und nur von der menschlichen Kreativität geschaffen werden können.

(Manfred J. Heinze/ Rolf Feis)



Dämonisches aus den Tiefen des Raumes

Nicht Texte, sondern perspektivische Projektionen räumlicher Objekte sind das Arbeitsgebiet des Editors „3-Demon“. Der Umgang mit den diversen Grafikprogrammen, wie Sculpt oder Videoscape wird dem User nun erleichtert.



Bequemes Editieren von räumlichen Objekten ist das Spezialgebiet von 3-Demon.

Glaubt man dem Hersteller, soll mit 3-Demon endlich ein 3D-Editor zur Verfügung stehen, der mit allen gängigen Grafikprogrammen arbeitet. Da für die Vielzahl dieser 3D-Programme kein Standardformat vorliegt, war es für die Programmierer sicher keine leichte Aufgabe, alle unter einen Hut zu bringen.

Wer die Diskette einlegt und das Programm über das Icon startet, wird erst einmal nach dem „so-und-so-vielten Wort“ aus dem „so-und-so-vielten“ Absatz aus dem 150seitigen Manual gefragt. Folgt man willig dieser Aufforderung, erscheint die Arbeitsfläche mit mehreren Zeichenbereichen, einigen Gadgets sowie einer umfangreichen Menüleiste, die sich nach Drücken der rechten Maustaste offenbart. Daß PAL für die Programmierer ein Fremdwort sein muß, zeigt sich an einem dicken, nutzlosen Farbbalken am unteren Ende des Bildschirmes. Das behindert nicht die Arbeit mit dem Programm. Es lassen sich dreidimensionale Drahtgitter-Objekte konstruieren, die dann mit einem beliebigen Ray-Tracing-Programm weiterverarbeitet werden. Das Programm unterstützt das Speichern von Objekten auf Diskette im Format von Sculpt-Animate, Videoscape, Silver, Forms in Flight, sowie Gosset Graphics (ein Name, der bisher noch nicht aufgefallen ist). 3-Demon wiederum versteht alle mit Videoscape und Sculpt-Animate produzierten Szenen. Dem Datenaustausch sind also wenig Grenzen gesetzt.

Nicht immer klappt der Transfer

Funktioniert das Schreiben von Videoscape-Dateien gut, so treten beim Einlesen hier und da Fehler bei der Farbgebung einzelner Flächen auf. So werden selbst einige Demo-Objekte von den Videoscape-Disketten nicht ordnungsgemäß geladen. Ein Problem ganz anderer Art ist die fehlende, auf das jeweilige Raytracing-Programm zugeschnitte, Objektoptimierung.

Alle Objekte bestehen aus Linien, Dreiecken oder Polygonen. Sie werden normalerweise in einem großen Fenster in perspektivischer Sicht sowie in einem kleineren von oben dargestellt. Im Gegensatz zu Sculpt-Animate, daß nur die Ansichten „oben“, „vorne“ und „seitlich“ erlaubt, läßt sich mit 3-Demon der Blickwinkel frei bestimmen.

Mit Eingabehilfen ist 3-Demon reichlich ausgestattet. Da gibt es Magneten, deren Wirkung sich mit der Entfernung zu jedem Punkte verringert, Funktionen zur automatischen Unterteilung von großen Flächen sowie Bewegungung — selbst von Teilen eines Objektes — um jede beliebige Raumachse. Beim letztgenannten Feature scheint das Programm in puncto Geschwindigkeit dem Sculpt-Animate-Editor hinterher zu laufen.

Alle entstandenen Flächen lassen sich mit den typischen Oberflächenformen versehen. So können Flächen transparent, metallisch, reflektierend oder glasig sein. Alle Attribute lassen sich mit 3-Demon komfortabel editieren. Dabei ist die Intensität der Effekte mittels Schiebereglern frei wählbar.

Als nützlich hat sich während der Testphase die Möglichkeit erwiesen, bestimmte Objekte als „Eltern“ und andere als „Kinder“ zu definieren. Damit ist eine Hierarchie gegeben, die sich zum Beispiel bei der Drehung von Objekten bemerkbar macht. Bewegt man ein Objekt einer höheren Hierarchieebene, drehen sich alle Niedrigeren mit — bewegt man eines einer niedrigeren Hierarchieebene, bleiben die „Eltern“ unberührt.

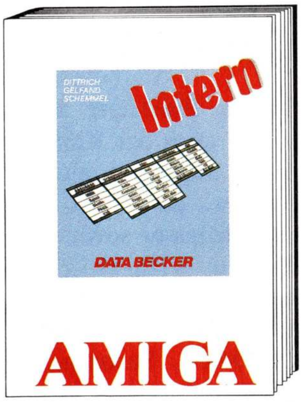
Die komfortablen Eingabehilfen von 3-Demon werden vor allem jene erfreuen, die nicht mit dem Sculpt-Editor arbeiten, der selbst ausreichende Möglichkeiten bietet.

(Ottmar Röhrig)

Info: DTM, Wiesbaden, Preis: ca. 200 Mark

1.3

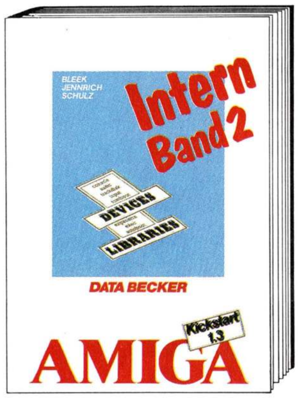
Heute lesen, was morgen passiert



Neuauflage: die Inside-Story.

Amiga Intern – ein Intern, wie man es von DATA BECKER gewohnt ist. Mit allem, was dazugehört: 68000-Prozessor, CIA, Blitter, Custom-Chips, die Strukturen von EXEC, I/O-Handhabung, Verwaltung der Resources, Erstellung eigener Devices, Exec-Base, resetfeste Programme, Autoboot mit der ROMboot. library, DOS-Funktionen, interne DOS-Bibliothek, Aufbau einer Diskette, Programmierung eigener DOS-Handler.

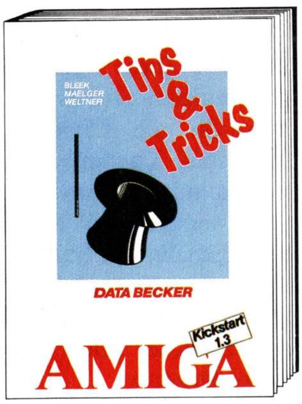
Amiga Intern
Hardcover, ca. 650 Seiten,
DM 69,–
erscheint ca. 9/88



Knallharte Informationen.

Amiga Intern Band 2 – das Buch für jeden aktiven Programmierer, der alle weiterführenden Informationen zu seiner Arbeit schnell und zuverlässig finden will: Ein- und Ausgabe über Devices, Standard-Austausch-Formate und Komprimierungsverfahren, alle Amiga-Libraries mit den dazugehörigen Strukturen, Basis- und Grundstrukturen, Preferences als Datenstruktur, Datenübermittlung von Workbench und CLI, Konventionen im Programmierstil und alles zur Version 1.3.

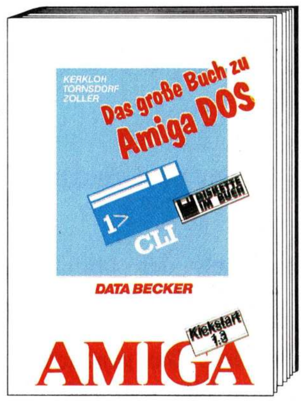
Amiga Intern Band 2
Hardcover, ca. 700 Seiten,
DM 69,–
erscheint ca. 9/88



Mit Programmen zaubern.

Mit Amiga Tips & Tricks läßt sich die Arbeit mit Ihrem Rechner noch effektiver gestalten – selbstverständlich bereits unter Berücksichtigung des neuen Betriebssystems (Version 1.3): Gestaltung eigener Programme, Tips & Tricks zum AmigaBASIC, Einbinden von Maschinenprogrammen in Amiga-BASIC, Einsatz von DOS-Routinen, optimierende Hilfsprogramme für AmigaBASIC-Programme, Tips zur Arbeit mit der Workbench, Aufbau der Icons, die neuen Preferences.

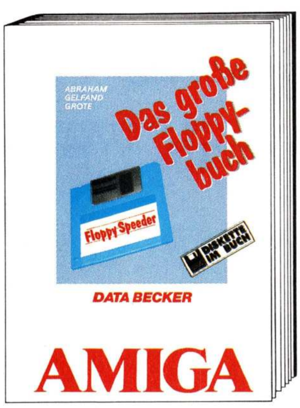
Amiga Tips & Tricks
Hardcover, 555 Seiten, DM 49,–



Runter von der Workbench.

Rein ins AmigaDOS: Umlenken der Ein- und Ausgabe, mit RAM-Disk und CLI arbeiten, STARTUP-Sequenz, Multitasking mit dem CLI, der interne Aufbau der CLI-Befehle, eigene CLI-Befehle programmieren... Das große Buch zu AmigaDOS – mit nützlichen Batch-Dateien und einer Beschreibung der neuen CLI-Befehle und Devices unter V1.3!

Das große Buch zu
AmigaDOS
Hardcover, 370 Seiten
inkl. Diskette, DM 59,–



Alles zur Amiga-Floppy.

Brandaktuell: die zweite, erweiterte Auflage mit allem, was Bezug zur Floppy hat: Workbench, CLI, Amiga-BASIC mit verschiedenen Dateitypen, Zugriff aufs Betriebssystem (mit File-Verwaltung, Trackdisk-Device, Boot-Block sowie Checksummen) und direkter Zugriff ohne DOS (MSM- und GCA-Codierung, Track lesen und schreiben, SYNC-Markierung). Dazu einen Floppyspeeder, einen Disketten-Monitor und ein schnelles, leistungsstarkes Kopierprogramm.

Das große Amiga Floppybuch
Hardcover, ca. 400 Seiten
inkl. Diskette, DM 59,–
erscheint ca. 9/88

COUPON!

HIERMIT BESTELLE ICH FÜR MEINEN AMIGA

NAME, VORNAME

STRASSE

ORT

zzgl. DM 5,– Versandkosten unabhängig von der bestellten Stückzahl
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

Dreimal Vielseitigkeit zum Thema Video

Mit multitalentierten Geräten will die Firma PBC dem Video-Desktop-Anwender die Handhabung erleichtern. Selbst die Tonspur des Videobandes ist in das Konzept eingebunden. Das Ganze ist hitverdächtig.



Hardware-Trio hilft bei Umsetzung des Computersignals

Zu einem Hit kann sich das neue PAL-RGB-Genlock von PBC entwickeln. Nicht nur Bildmischer für Computer- und Videosignale, bearbeitet es auch Tonsignale und kann zur Nachvertonung eigener Filmsequenzen professionell eingesetzt werden. Ein Überblendregler für das Mikrofon fehlt ebensowenig, wie Anschlüsse für externe Tonquellen mit Normpegel (CD, Tape, Tuner). Am Ausgang steht dann für Audioaufnahmen das abgemischte Signal zur Verfügung.

Jedes eingespeiste FBAS-Signal kann in weiten Bereichen mittels der üblichen Regler Farbsättigung, Kontrast und Helligkeit verändert werden. Das gemischte Signal von Video- und Computerquelle liegt am Ausgang als RGB- sowie als FBAS-Signal an. So ist gesichert, daß die benötigte Signalform vorhanden ist.

Multiprozessor ersetzt Farbscheiben

Ein weiterer Allround-Könner ist der PAL-RGB-Multiprozessor. Es ist ein RGB-Farb-Splitter, der aus einem FBAS-Signal jeweils den Rot-, Grün- und Blauanteil herausfiltert. Dort kann dann ein Digitizer angeschlossen werden, der normalerweise mit den drei entsprechenden Farbfilterscheiben betrieben werden müßte. Durch die Regelmöglichkeiten bei den einzelnen Signalen, werden so wesentlich bessere Ergebnisse erzielt, als bei der Verwendung von Farbfilterscheiben. Der ebenfalls integrierte Video-Color-Prozessor dient der nachträglichen Veränderung eingespeister FBAS-Signale. Damit sind nicht nur Korrekturen, sondern auch gezielte Verfremdungen möglich. Wird das bereits gemischte Genlocksignal derart weiterverarbeitet, erhält man nahezu unendliche Variationsmöglichkeiten.

Bei der Abspeicherung auf Videoband ist der RGB-PAL-Umwandler behilflich. In dieser Komponente des RGB-PAL-Multiprozessors wird ein RGB-Signal eingespeist, welches am Ausgang als Video-tüchtiges FBAS-Signal zur Verfügung steht. Der Vorteil gegenüber dem FBAS-Ausgang des Amiga 1000 liegt hier in der wesentlich höheren Qualität des Ausgangssignales. Für 898 Mark erhält man ein für die Zukunft gerüstetes Gerät, das nicht nur mit dem oben angeführten PAL-RGB-Genlock zufriedenstellend arbeitet.

Als Geheimtip scheint sich das dritte Gerät der PBC-Reihe zu entpuppen. Für 698 Mark erhält man einen Realtime-Video-Digitizer, der in allen Modi des Amiga arbeitet. Mittels der komfortablen Software sind mannigfaltige Verfremdungen des digitalisierten Bildes möglich, ohne daß eine Neudigitalisierung erfolgen muß.

Komplettpaket löst Haarprobleme

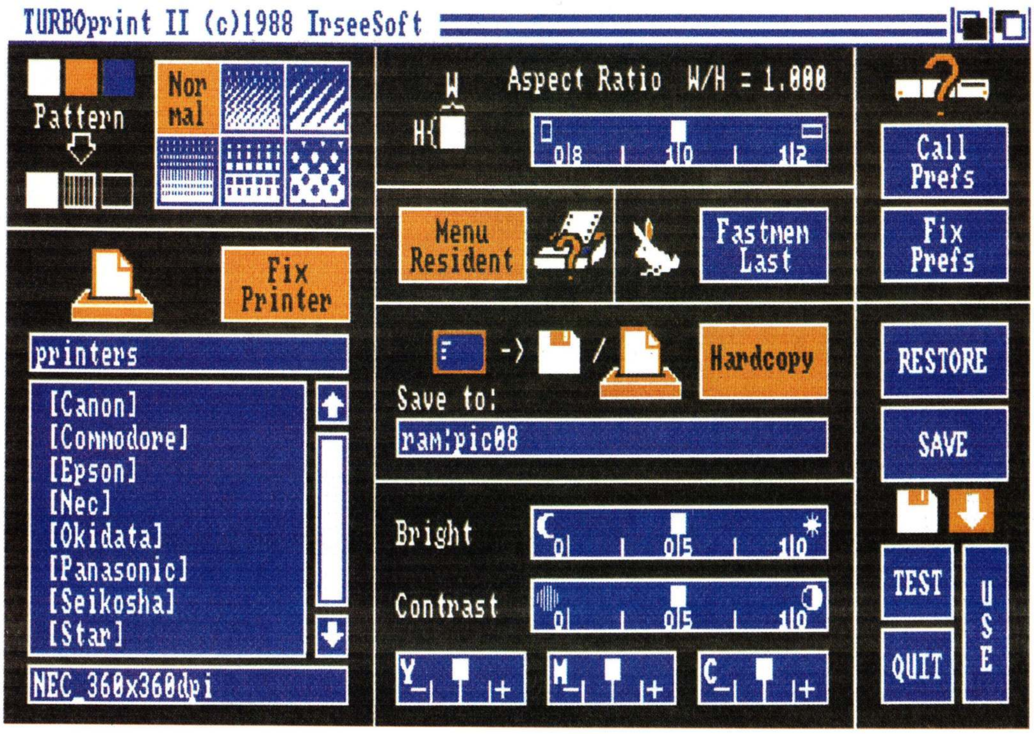
Für den Einsteiger interessant ist auch das Komplettpaket, bestehend aus dem Digitizer, dem PAL-RGB-Multiprozessor sowie einem digitalen VHS-Videorecorder hoher Qualität, für einen angemessenen Preis von 2598 Mark. Sozusagen als Nebenprodukt bei der Entwicklung dieser Geräte kam es zum „Hairstyle-Vario-System". Diese spezielle Branchenlösung für Friseure ermöglicht die Digitalisierung eines Kundenkopfes, dem dann auf dem Bildschirm verschiedene Perükken angepaßt werden können. In diesem Komplettpaket fehlen weder Thermotransferdrucker noch Schulungsprogramm.

(Ottmar Röhrig)

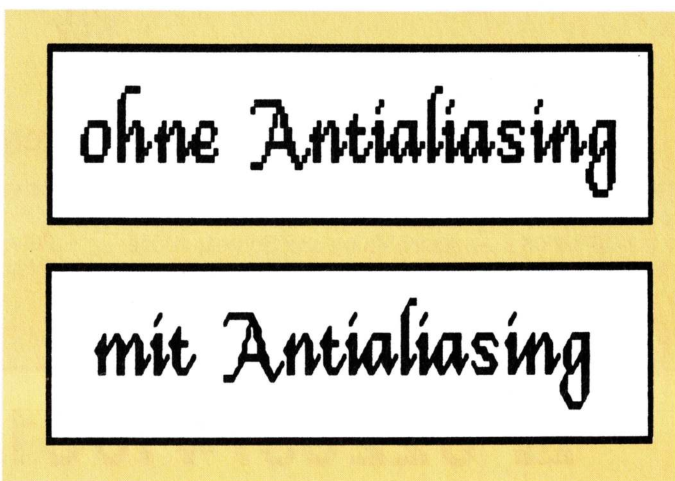
Info: PBC Peter Biet Computertechnik, Georg-Fischer-Str. 5, 6415 Petersberg 2, Tel.: 06 61/60 12 63

TURBOprint II optimiert den Drucker

Noch ist die Workbench 1.3 nicht veröffentlicht, schon existiert das erste Programm, um die Druckroutine aufzubessern. „TURBOprint II" versetzt den Drucker in einen Geschwindigkeitsrausch.



Das TURBOprint II-Hauptmenü gibt einen Eindruck von der Vielseitigkeit des Programms.



Notepad-Text mit und ohne Glättefunktion

Die völlig überarbeitete Version 2.0 von TURBOprint II verspricht hohe Druckqualität und Schnelligkeit. Das neue Einstellmenü läßt erkennen, daß Wesentliches geändert und hinzugefügt wurde. Ein einziger Menübildschirm reicht nicht mehr für alle Funktionen aus. Viele Einstellungen müssen deshalb auf einer zweiten Kontrollebene ausgeführt werden. Jetzt lassen sich Helligkeit und Kontrast des Bildes nachregulieren. So kann man mit jeder Farbbandqualität optimal arbeiten: Altersschwache Farbbänder werden durch Nachregulierung wieder farbstark. Grafiken, speziell digitalisierte HAM-Bilder mit 4096 Farbtönen, waren farbstichig, dunkel oder kontrastarm. Abhilfe schaffen Farbregler für die drei Grundfarben, mit denen jede Mischfarbe korrigiert werden kann. Bei Schwarzweiß-Ausdrucken wirken diese Regler auf die Helligkeit der jeweiligen Farben im Endergebnis. Auch die neue Workbench wird diese Möglichkeit bieten, jedoch der Effekt ist minimal (beim Mandrill mit NEC P6 Color sind kaum Unterschiede zu erkennen). Grafiken und Schriften erschienen bisher grob aufgelöst auf dem Papier. Hier wird die Glättefunktion (Antialiasing) von TURBOprint II eingesetzt. Wie der Vergleich zeigt, werden die Buchstaben ansehnlich, störende Treppen verschwinden.

Diese Funktion gibt es auch bei der Workbench 1.3, doch ist TURBOprint II deutlich schneller (siehe Tabelle).
Neu hinzugekommen sind zusätzliche Formateinstellmöglichkeiten in 1/10 Zoll oder in Druckpunkten. Ausdruckgrößen können außer in Zoll auch in Millimetern angegeben werden.
Um sowohl mit 90×90 Pixel/Zoll, als auch mit der 360×360-Punkte-Auflösung vernünftige Ergebnisse zu erzielen, gibt es jetzt insgesamt sechs verschiedene Raster zur Farbumsetzung. Bei der niedrigen Auflösung empfiehlt sich ein sehr dichtes Muster, um einen satten Ausdruck zu erzielen. Bei der Höchstauflösung wiederum produziert das Muster zu dunkle, unbrauchbare Flecken. Hierfür verwendet man am besten ein sehr grobes Muster, damit nicht zuviel Farbe aufgetragen wird (siehe Abbildungen). Damit lassen sich interessante Effekte wie zeitungsbildähnliches Aussehen erzielen.
Eine weitere Neuerung ist das integrierte Festplatten-Installationsprogramm.
TURBOprint II kann nun jederzeit nachgeladen werden (auf Tastendruck oder über Icon). So entfällt das Abspeichern vor dem Druck. Das Einstellmenü kann resetfest eingebunden werden und steht auf Tastendruck jederzeit für schnelle Änderungen vor dem Drucken zur Verfügung. Die Idee, das ganze Menü ohne Nachladen erscheinen zu lassen, kostet allerdings 28 KB Speicher.
Die resetfeste Hardcopy-Funktion wurde erweitert. Bilder im IFF-Format lassen sich jetzt auf ein beliebiges Speichermedium — RAM-Disk, Festplatte oder Diskette — speichern. Die Bilder können in ein Malprogramm geladen werden.
Da sich bei vielen Malprogrammen keine Ausschnitte drucken lassen, haben die Entwickler in Anlehnung an Deluxe Paint eine Fadenkreuz-Schere programmiert, mit der beliebige Ausschnitte bestimmt und ausgedruckt oder gespeichert werden. Laut Hersteller arbeiten die WB-1.2-kompatiblen Treiber in der Version II um 40 Prozent schneller. Bei hohen Auflösungen sind die mitgelieferten Turbo-Treiber im Vergleich mit den bisherigen mehr als doppelt so schnell. Alle Treiber zusammen benötigen 3 bis 4 KB Speicher. Wie aus der Tabelle ersichtlich, werden für alle gängigen Drucker — inklusive Laserdrucker — Treiber mitgeliefert. Weiterhin vorhanden ist der NoFastMem-Schalter, der es ermöglicht, auf einem Amiga mit Speichererweiterung Programme zu verwenden, die sonst nur bei abgeschalteter Erweiterung laufen. Laut Hersteller können die zahlreichen Besitzer von TURBOprint II ein Update mit neuem Handbuch direkt von IrseeSoft beziehen.

Info:
IrseeSoft SPCS, Grüntenstr. 6, 8951 Irsee, Preis: 98 Mark



24-Nadeldrucker-Farbgrafik mit TURBOprint II gedruckt

TURBOprint II: Unterstützte Drucker

Hersteller	Drucker
Canon	*PJ-1080A*
	BJ-130
	LBP-8II
Commodore	*MPS 1500*
	MPS 2000
Epson	*LX/FX/EX 800/1000*
	FX 80, JX 80
	LQ/SQ
	GQ 3500
HP	*Laserjet+/Laserjet II*
NEC	*P5*
	P6, P6 plus
	P7, P7 plus
	P9
Okidata	*ML 194, 294*
	ML 380, 391, 393
	Okimate 20
Panasonic	*KX-P 1081, 1083, 1595*
	KX-P 1180, 1195, 1695
	KX-P 1124, 1540
	KX-P 4450
Seikosha	*MP 1300AI*
	SL 80 AI/IP
Star	*NL 10*
	LC 10
	NB 24-10/15
	LaserPrinter 8

TURBOprint II: Geschwindigkeitsvergleiche

	TURBOprint II	*WB 1.3*
EPSON LQ 2550 180×180 dpi		
DIN-A5-Grafik		
Hardcopy S/W	*1:14*	*1:23*
mit Antialiasing	*1:19*	*3:29*
COLOR-Grafik	*1:13*	*2:09*
Laserdrucker:		
Star Laserprinter 8		
Hardcopy DIN-A5		
300×300 dpi	*1:40*	*3:52*
+ Antialiasing	*2:15*	*8:20*



Illustriert von Sandra Filippucci

Einkaufsliste für Videografen

Es wird ständig neue Hard- und Software entwickelt und angeboten, die Grafik- und Animationsfähigkeiten des Amiga voll ausnutzt. Sowohl in professioneller Umgebung als auch im Heimbereich stehen Tools zu erschwinglichen Preisen zur Verfügung. AmigaWelt hat das umfassende Angebot von Hardware- und Software-Tools durchkämmt, mit denen sich der Amiga in ein Grafikanimationssystem nach dem neuesten technischen Stand verwandeln läßt.

SOFTWARE

VIDEO-TITEL-PROGRAMME

Titelgeneratoren sind heutzutage aus der Fernsehproduktion nicht mehr wegzudenken. In Fernsehstudios werden sie für verschiedene Zwecke eingesetzt, zum Beispiel beim Abrollen von Informationen vor einem Teleprompter (Teleprojektor) oder zum Einblenden von Bestellhinweisen am Ende einer Produktwerbung. Vereinfacht ausgedrückt, handelt es sich bei dem Titelgenerator um ein Tool, mit dem sich Text erzeugen und in Form von Titeln und weiteren Informationen auf Video übertragen läßt. Der Amiga ist ein hervorragender Titelgenerator, da er über ein Genlock-Interface an externe Videoquellen angeschlossen werden kann. Er eignet sich ebensogut zum Einsatz im Heimbereich wie im Studiobereich.

PRO VIDEO CG1 (ca. 400 Mark). Der Titelgenerator Pro Video CG1 der Firma PVS Publishing wird mit neun verschiedenen Fonts geliefert, die allesamt gleichzeitig verwendet werden können, wobei sich aus den 16 der möglichen 4096 Farben ein Farbton auswählen läßt. Jede Textzeile ist in einem anderen der 13 Schattentypen darstellbar. Maximal 100 Textseiten können für eine unmittelbare Bearbeitung im RAM gespeichert werden. Für Seiten- und Bildschirmwechsel stehen die Überblendeffekte Rolling, Revealing, Pulling, Wiping, Slides, Checkerboard und Crawls mit diversen Geschwindigkeiten und Verweilzeiten zur Verfügung. Dieses Programm ist ebenfalls in der PAL-Version erhältlich.

PRO VIDEO PLUS (ca. 500 Mark), ebenfalls von PVS Publishing, umfaßt zusätzlich zu der breiten Feature-Palette von Pro Video CG1 vier interne Fonts sowie Grafiken. Als beliebte Zusatzfunktion wurde in dieses Programm die Fähigkeit integriert, IFF Bilder als Hintergrund einzuladen, um sie mit Text zu überlagern. Mehr als 90 Seitenübergangsformen, darunter Rolls, Crawls, Slides, Wipes, Reveals und zeilenunabhängige Bewegungsmethoden, wurden integriert. Dieses Programm, das ähnlich aufgebaut ist wie sein Vorgänger, arbeitet speicherabhängig und erfordert mindestens 1 MB zusätzliches Fast RAM.

VIDEO TITLER (ca. 250 Mark) von der Aegis Corporation ist ein weiteres Software-Paket für die Titelerzeugung, mit dem sich sowohl Zeichen als auch Spezialeffekte realisieren lassen. Das Programm arbeitet im 16-, 32- oder 64-Farben-Modus (HALFBRIGHT). Für einen schnelleren Zugriff können den Funktionstasten bis zu zehn Fonts gleichzeitig zugeordnet werden. Darüber hinaus sind Fonts im Expert-Modus für die unterschiedlichsten Effekte variierbar. Der Video Titler verfügt über einen Spezialeffekt-Modus für verschiedene Tricküberblendungen und Änderungen der Anzeigeform. Der Leistungsumfang wird durch das unterstützte ANIM-Format für komprimierte Animation und durch True Overscan (768 x 480) abgerundet.

TV TEXT und **TV SHOW** (jeweils ca. 190 Mark), zwei von Brown-Wagh Publishing angebotene Programme, sind, obwohl sie zusammen eingesetzt werden, auch einzeln erhältlich. Mit diesen Programmen können Spezialtitel kreiert und auf Video übertragen werden. Mit TV Text werden Titel in einem der neun möglichen Video-Fonts erzeugt. Mit TV Show lassen sich die mit TV-Text kreierten Titel sowie andere IFF-Brushes vor einem IFF-Hintergrund mit der Flying- oder Wiping-Technik ein- und ausblenden.

STATION MANAGER SERIES (ca. 295 $) Der letzte Titelgenerator, der an dieser Stelle erwähnt werden sollte, gehört ▶

zum kompletten Softwarepaket von Associated Computer Services. Jedes Programm kann jedoch auch einzeln erworben werden. Das erste Programm dieser Reihe ist der Station Manager/Character Generator, ein Programm für die Titelerzeugung mit vollem Leistungsumfang. Der Station Manager/Teleprompter ist ein Teleprompter-Emulator mit Sendequalität und daher für die Nachrichtenübertragung geradezu ideal. Mit Station Manager/Weather Graphics lassen sich nicht nur Wettergrafiken für Fernsehstudios erzeugen, es können zudem Informationen von Wetterstationen automatisch eingeladen werden.

ANIMATIONSPROGRAMME

Überall begegnet man 3D-Grafiken: Bei Sportübertragungen, Werbespots und bei Kino- und Fernsehfilmen. Infolge der jüngsten Entwicklung auf dem Gebiet der 3D-Wiedergabe- und Animationssoftware, die über die vielfältigsten Sonderfunktionen verfügt, öffnet sich die Welt der 3D-Animation nun auch für Benutzer von Personalcomputern. Wegen der Komplexität der Animationsprogramme und ihres hohen Speicherbedarfes empfehlen die Hersteller einen Mindest-Speicherumfang von 1 MByte Fast RAM.

SCULPT 3D (ca. 160 Mark) von Byte by Byte arbeitet nach der Ray Tracing Methode, mit der realistische Objektoberflächen und Schatteneffekte zu erzielen sind. Beim Ray Tracing werden eine oder mehrere Lichtquellen auf einen Punkt gerichtet, während der Computer jeden Lichtstrahl bis zu seinem Ende abtastet. Es lassen sich Objekte mit verschiedenen Oberflächenformen darstellen, vom Weichkristall-Typ (smooth crystal) bis zur gefurchten Panzerstruktur eines Gürteltieres (bumpy armadillo shell). Sculpt 3D arbeitet im HAM-Modus und unterstützt Overscan (704 × 480). Leider besitzt dieses Programm nicht die Fähigkeit, Bilder zu animieren. Es ist nicht kopiergeschützt und Genlock-fähig.

ANIMATE 3D (ca. 250 Mark), ebenfalls von Byte-by-Byte, erlaubt das Einladen von Objekten, die mit Sculpt 3D erzeugt worden sind, sowie das unabhängige Kreieren von Objekten. Mit Animate 3D können Licht-, Kamera- und Objektbewegungen über ein grafisches Interface oder mit Hilfe einer einfachen Script-Sprache gesteuert werden. Bilder im IFF-Format können als Vorder- beziehungsweise Hintergrund dienen, dazwischenliegende Objekte lassen sich animieren. Die Animationen sind im Drahtgitter-Modus zu betrachten oder als komprimierte Animations-Files ähnlich der Files im Anim-Format abzuspeichern. Sie können auch zum Anzeigen im Frame Buffer gespeichert werden. Animate 3D ist nicht kopiergeschützt und unterstützt Overscan (704 × 480) sowie das Genlock-Interface.

SILVER (ca. 225 Mark), ein Programm der Firma Impulse Inc., fällt ebenfalls in den Bereich Ray Tracing — Animation. Es faßt die Funktionen von Sculpt 3D und Animate 3D in einem Paket zusammen. Obwohl mathematisch nicht so leistungsfähig wie Sculpt 3D, zeichnet es sich durch eine höhere Geschwindigkeit in der Bilddarstellung aus.

Ein Wort zu Overscan

DESKTOP VIDEO wäre ohne Overscan nicht denkbar. Obwohl dieses Feature nicht zu den Standardfunktionen des Amiga zählt, besitzt ein mit dieser Funktion ausgestattetes Programm die Fähigkeit, den gesamten Video-Bildschirm auszufüllen, ohne eine Umrandung zu hinterlassen. Leider hat sich für diese Funktion noch kein Standard durchgesetzt. So arbeitet zum Beispiel VideoScape 3D in der Interlace-Bildschirmgröße 352 × 440, was die untersten 20 Bildschirmzeilen am Monitor als schwarzen Balken erscheinen läßt. Dies ist ein Handicap für professionelle Anwendungen, wenn es sie nicht sogar völlig ausschließt (Aegis ist sich dieses Problems bewußt und stellt eine Verbesserung für die Update-Version in Aussicht). Im HiRes-Modus von Deluxe Paint II wird man mit einem ähnlichen Problem konfrontiert. In der Auflösung 672 × 444 wird das Bild mit einem Rahmen umzogen, der es für eine Fernsehübertragung unbrauchbar macht.
Als Idealauflösungen, mit denen zur Zeit einige Programme arbeiten, gelten 384 × 460 Interlace und 768 × 480 HiRes. Dennoch stellt dieser Balken bei Titelgeneratoren nicht unbedingt ein Problem dar, da sie über die Funktion „Safe title area“, etwa „geschützte Titelzone“, verfügen, welche bewirkt, daß Zeichen nicht außerhalb des Bildschirms plaziert werden können. Somit ist Overscan mit einer Auflösung von 672 × 444 Bildpunkten für diese Anwendung durchaus akzeptabel, doch bei Mal- und Animationssoftware sei der Nutzen dieser Funktion in Frage gestellt.

VIDEOSCAPE 3D (ca. 260 Mark) von Aegis Development legt eine Animationsmethode zugrunde, die als „solid modeling“ bezeichnet wird. Sie gestattet das Arbeiten mit Licht- und Schatteneffekten auf den Oberflächen. Objekte werden auf ähnliche Weise wie bei einem Zeichner erzeugt. Jede Linie wird mit einem Plotter auf Zeichenpapier gebracht, und anschließend werden die Informationen in einem Script-File festgehalten, den das Programm für die eigentliche Realisierung des Objektes benötigt. Drei Programme zum Erzeugen und Editieren von Objekten werden mitgeliefert (zur Zeit wird ein Programm von Aegis Beta-getestet, das die Arbeit weiter erleichtern soll). Lichtquelle, Kameraführung und Objektbewegung werden mit Hilfe der Script-Files gesteuert. Die Echtzeit-Wiedergabe der Animation wird im ANIM-Format, einer komprimierten Animationsmethode, bewerkstelligt. Das Paket ist nicht kopiergeschützt; es unterstützt das Genlock-Interface und eingeschränktes Overscan in allen Auflösungsstufen (Aegis arbeitet an einer Update-Version, die True Overscan integriert).

ANIMATOR: APPRENTICE (ca. 190 Mark) ist ein Programm der Fa. Hash Enterprises, mit dem sich 3D-Figuren animieren lassen. Das Paket ist modular aufgebaut, wobei das erste Modul bei der Objektgestaltung Bilder aus einem Malprogramm oder mit einem Digitizer erstellte Bilder verwendet. Mit der Vorder- und Seitenansicht eines Objektes läßt sich die dreidimensionale Ausführung realisieren. Anschließend bietet das Modul die Möglichkeit, ein digitalisiertes Bild (zum Beispiel ein Gesicht) zur realitätsgetreueren Darstellung über die Objektmaske zu „schieben" (matte). Die weiteren Module erlauben eine vielfältige Manipulation des Objekts beziehungsweise der Objekte.
Das Programm arbeitet im HAM-Modus und gestattet somit optimale Schattierungen und realistische Farben. Weitere Pluspunkte sind die justierbare Lichtquelle und die perspektivische Kameraführung. Zum Lieferumfang gehört eine vordefinierte Bibliothek mit Muster-Bewegungsabläufen und Figuren. Animator: Apprentice unterstützt Overscan (352 × 240 HAM und 704 × 480) und das Genlock-Interface. Ein Kopierschutz liegt nicht vor.

FORMS IN FLIGHT (ca. 130 Mark), mit diesem Programm von Micro Magic lassen sich sowohl 2D als auch 3D-Objekte kreieren. Über Pull-down-Menüs können Features wie Kameraschwenkungen, Betrachtungspunkte, Lichtquellen, Vergrößerungen, Schatteneffekte, Editing usw. leicht gehandhabt werden. Der Programmteil für die Bilderzeugung unterstützt die Auflösungsstufen 640 × 400 und 640 × 200 mit jeweils 16 Farben. Die Wiedergabe der Sequenz ist in allen Auflösungsmodi möglich; im LoRes und Interlace-Modus in bis zu 32 Farben. Das Fast Flight Player Programm gestattet die Echtzeit-Wiedergabe mit Overscan (704 × 444). Das Programm ist nicht kopiergeschützt; einige Demo-Objekte sind mitenthalten.

DELUXE PRODUCTIONS (ca. 340 Mark) von Electronic Arts. Bei diesem Programm handelt es sich um ein Grafik-Animationstool in HiRes, das besonders für 3D-Business-Anwendungen hervorragend geeignet ist. Für eine ruckfreie Animation sorgt double buffering, über 40 schnelle Bildübergangsarten sind integriert worden, zum Beispiel Spiraldrehung, Diagonalüberblendung sowie der Jalousie- und Streuungseffekt (Scatter). Sie verleihen dem Programm alles in allem einen professionellen Touch.
Das Programm integriert den Overscan-Modus und ist IFF-kompatibel. Weiterhin verfügt es über hochauflösende Hintergründe, die sich insbesondere bei Sportübertragungen, Nachrichten und Wetterberichterstattungen anbringen lassen, sowie bei Business-Grafiken und Fonts. Deluxe Productions besitzt einen Kopierschutz und ist Genlock-kompatibel.

MALPROGRAMME

Bei Kino- und Fernsehfilmproduktionen werden oft Szenen gedreht, für die es zu umständlich oder gar unmöglich wäre, ein aufwendiges Szenenbild anzufertigen. Das ist zum Beispiel bei Phantasy- und Science-Fiction-Filmen oft der Fall. Hier läßt der Regisseur den Künstler ein Maskenbild (Matte) entwerfen; ein Teil des Szenenbildes, zum Beispiel ein Fenster, wird aufgebaut, und der restliche Bildinhalt wird als Maske um das Set gezeichnet beziehungsweise gemalt. Zum Beispiel aus einem Hochhaus-Hintergrund in einer Utopie-Stadt oder einem Mond über einem fremden Planeten bestehen. Mit einem Malprogramm und einem Genlock-Interface kann solch ein Bild in den eigenen vier Wänden realisiert werden.

DELUXE PAINT II (ca. 200 Mark), das beliebte Malprogramm von Electronic Arts, enthält Features, die sonst nur Paint-Boxen höherer Preiskategorien aufzuweisen haben. In allen vom Amiga unterstützten Auflösungsmodi mit Ausnahme des HAM-Modus lassen sich mit Deluxe Paint II frei nach Gusto und Phantasie die komplexesten Bilder zaubern. Es ist für Genlockbetrieb geeignet und hat einen Kopierschutz.

DIGI-PAINT (ca. 100 Mark). Bevor NewTek Inc. Digi-Paint herausbrachte, wurde der Hold-and-Modify-Modus (HAM) des Amiga von keiner Software voll ausgenutzt. Dieses Programm ermöglicht den Zugriff auf alle 4096 Farben des Amiga sowie auf einige weitere Farben über den Raster-Modus (Dithering). Ein hervorragendes Feature ist die Einfärbe-Funktion (das Einfärben von Schwarzweißbildern in jeder beliebigen Farbschattierung).
NewTek beabsichtigt, die Overscan-Funktion in zukünftige Update-Versionen zu integrieren. Obschon Digi-Paint nicht kopiergeschützt ist, muß für den Zugriff auf das Programm ein Paßwort eingegeben werden. Das Genlock-Interface wird unterstützt.

PRISM (ca. 160 Mark) und PRISM PLUS (ca. 120 Mark) von der Firma Impulse Inc. sind zwei weitere HAM-Editoren für den Amiga. Prism! ist für Computer mit 512 KByte ausgelegt, während Prism Plus einen Speicherumfang von 1 MByte voraussetzt. Beide Programme gehören zum Lieferumfang von Prism Plus. Es lassen sich Bilder in bis zu 4096 Farben erstellen und editieren; zudem können Bilder in einer Auflösung von 1024 × 1024 Bildpunkten erzeugt werden.

PROGRAMME UND UTILITIES

Diese Programme nehmen eine Sonderstellung ein, da sie andere Programme bei der Durchführung spezieller Aufgaben unterstützen.

INTERCHANGE (ca. 75 Mark), ein Programm der Fa. Syndesis, sorgt dafür, daß Objekt-Files von VideoScape 3D und Sculpt 3D gemeinsam verarbeitet werden können. Objekt-Files, die mit dem Objekteditor in Sculpt 3D angefertigt worden sind, lassen sich in ein Format umwandeln, das eine Weiterbearbeitung und Animation mit VideoScape 3D gestattet. Umgekehrt können auch mit VideoScape 3D kreierte Objekte von Sculpt 3-D in Ray-Tracing-Szenen eingebunden werden.

THE DIRECTOR (ca. 100 Mark) von der Right Answers Group ist ein Composite-Sequencing-Programm mit der Fähigkeit, ganze IFF-Bildschirme oder Ausschnitte davon zu klappen, und zwar in allen unterstützten Auflösungsstufen sowie

in Overscan. Es lassen sich Animationen aus Video-Scape 3D und digitalisierte Klangfolgen wiedergeben. Weiterhin bietet es simultanes Ein- und Ausblenden, Farbanimation (Color Cycling), sowie die Überblendeffekte Dissolving und Wiping.

ANIMATION EFFECTS (ca. 100 Mark), ein Programm der Fa. Hash Enterprises, wurde zur Simulation von Digitalen Videoeffekten (DVE) bei gleichzeitiger Unterstützung von Overscan geschrieben. Es bietet dem Benutzer die Möglichkeit, einen Teil eines IFF- oder HAM-Bildes „auszuschneiden" und mit den Funktionen klappen und drehen und mit Hilfe der Flying-Technik zu animieren. Es arbeitet auf ähnliche Weise wie ein ADO (Ampex Digital Opticals, ein Gerät für Überblendungen und Spezialeffekte).

PAGEFLIPPER (ca. 100 Mark), ein von Mindware International angebotenes Programm, erlaubt das schnelle „Durchzeigen" von beliebigen IFF-Bildkombinationen in allen Auflösungsstufen. 30 Einzelbilder werden in der Sekunde angezeigt und erzeugen so einen Bewegungsablauf wie bei der frühen Folien-Animation. Das Programm verbraucht den gesamten verfügbaren Speicherplatz bis hin zu 8,5 MByte. Overscan (764 × 480) wird unterstützt.

BUTCHER (ca. 80 Mark). Mit dem Programm Butcher der Firma Eagle Tree Software können Bilder bearbeitet werden, die mit einem Digitizer oder einem Malprogramm angefertigt wurden. Ein Bild läßt sich von einem Auflösungsmodus in einen anderen konvertieren, unter anderem auch in den HAM-Modus. Als weitere Funktionen sind Positiv/Negativ-Umkehrung, Mosaik-Darstellung, Farbtrennung und Echtes Overscan zu nennen.

PIXMATE (ca. 120 Mark). Dieses Programm der Firma Progressive Peripherals & Software ist ein Bildbearbeitungsprogramm, mit dem sich Bilddetails noch exakter gestalten lassen. Es können ebenfalls diverse Spezialeffekte erzeugt werden. Overscan und Halfbright-Modus (64 Farben) sind möglich; Bilder lassen sich in andere Auflösungsstufen konvertieren.

HARDWARE

GENLOCKS

Der Amiga ist der erste Genlock-fähige Home-Computer, daß heißt, seine Signale können mit denen einer externen Videoquelle (Videokamera, Videorecorder und so weiter) „synchronisiert" und kombiniert werden. Um zu verstehen, wie Video funktioniert, stelle man sich vor, wie eine Filmkamera einen Bewegungsablauf aufzeichnet. Jedes einzelne Bild ist durch eine bestimmte Größe gekennzeichnet, die horizontal und vertikal festgelegt ist. Das gleiche gilt auch für Videobilder. Für jedes Einzelbild gibt es einen Anfangs- und einen Endpunkt; diese Information ist im Synchronsignal enthalten. Die Signale aller Video-Komponenten müssen synchron verlaufen, wenn sie zusammen verwendet werden sollen. An dieser Stelle erscheint das Genlock auf der Bildfläche. Mit dem Genlock-Interface können Computergrafiken über ein Video gelagert werden (Keying); das auf diese Weise entstandene Bild wird als Endbild aufgezeichnet.

COMMODORE A-1300. Das erste Genlock-Interface auf dem Markt, das Commodore A-1300, ist ausschließlich für Hobby-Filmer konzipiert. Für Fernsehübertragungen ist es weniger geeignet. Es synchronisiert die Bildsignale des Computers mit denen einer externen Video-Quelle. Mit einem Umschalter können drei Positionen eingestellt werden, die bestimmen, wie die Videosignale verändert werden. In der ersten Position werden die Signale des Computers unverändert übermittelt, die zweite Position gibt die eingehenden Videosignale unverändert aus, und in der dritten Position werden Grafiken über das eingehende Video gelagert (Keying).

SUPER GEN. Dieses Genlock wurde von einem Ingenieur der Ex-Grass Valley Group ausschließlich für den Amiga entwickelt und wird von der Fa. Digital Creations angeboten. Es verfügt sowohl über einen Video-Eingangs- als auch einen durchgeschliffenen Anschluß, sowie über zwei Video-Ausgänge, ein Key-Signal (Austastsignal) für Keyer in Bandlaufrichtung und zwei Fader-Regler. Der erste Fader regelt die Schwärze der eingehenden Videosignale, der zweite löst Amiga-Grafiken über dem Video auf (dissolve). Beide Fader sind sowohl über die Software als auch manuell steuerbar. Das Programm Pro Video Plus von PVS Publishing enthält die für diese Fader notwendigen Treiber. Im Lieferumfang ist ebenfalls ein Notchfilter (Fallenfilter) zum Ausschalten von Chroma Cross talk inbegriffen. Die Video-Ausgänge sind durch NTSC-Rundfunkqualität gekennzeichnet und entsprechen der amerikanischen Rundfunknorm RS 170A. Alle Anschlüsse entsprechen der BNC-Norm für professionelle Anwendungen und sind auf einfache Weise an Sendegeräte anzuschließen.

AMIGEN. Das Amigen der Fa. Mimetics Corporation ist ein Low-Cost-Genlock, das Composite-Video-Signale empfängt und sie mit Computergrafiken überlagert. Die gemischten Signale zeichnen sich durch Sendequalität aus.

GENKEY. Zum Lieferumfang von SciTech's GenKey gehört ein Genlock, ein Time-base-corrector (TBC) und ein Keyer, alles in einem Gerät. Es enthält einen Schalter für die vertikalen Intervalle zum reibungslosen Umschalten zwischen der externen Videoquelle, Computer-Video und dem Key-Video. Ein Video-Korrekturverstärker wurde in das Paket eingebaut, mit dem die Sättigung der verschiedenen Farbkomponenten sowie die Schwarzabhebung (Setup-level) justiert werden kann. GenKey ist für externe Videosignale durchgeschliffen und besitzt einen Ausgangswiderstand von 75 Ohm; ein RGB-Ausgang und ein Composite-Video-Ausgang mit Sendequalität sind vorhanden. Alle Anschlüsse entsprechen dem BNC-Standard für professionelle Anwendungen, was den Anschluß an Sendegeräte vereinfacht.

RM-2. GlennLoc's Synch-Generator mit Genlock besitzt die Fähigkeit, Signale einer externen Videoquelle zu empfangen und sie auf die RS-170A-Ausgangsnorm zu synchronisieren.

RM-2 wird in einem 19-Zoll-Gehäuse für Video-Racks geliefert. Die Subcarrier-Phase (Chrominanzträgerphase) und die horizontale Phase lassen sich extern justieren, ein interner Farbbalken-Generator ist ebenfalls vorhanden. Weiterhin ist die Funktion implementiert, den Videopegel von Grafiken zu steuern, die mit den externen Videosignalen gemischt werden.

VCG-3 (ca. 2000 Mark). Ein kompaktes Genlock-Interface der Firma Videocomp, für den professionellen Einsatz am Betacam oder U-Matic Schnittplatz. Die Bandbreite der RGB-PAL-Wandlung umfaßt 5,5 MHz. Stufenloses Fading und Superimposing (manuell oder automatisch) sowie vorwählbare Fade-In/Out-Zeiten und eine Inverse-Schaltung sind die auffallendsten Bestandteile des integrierten Videomischers. Das VCG-3 verfügt über zwei Videoeingänge (1 BNC, 1 RGB analog), sowie drei Videoausgänge (2 BNC, 1 RGB-analog).

VCG-3 P (ca. 3000 Mark). Die professionelle Ausführung des VCG-3, mit Down Stream Keyer, Black-Burst-Input sowie einer auf die Studiosynchronisation einpegelbare Farb- und Horizontalphase.

AG-5 (ca. 1000 Mark). Das kleinste und günstigste Gerät der Genlock-Serie von Merkens Computer Video Systeme. Dieses handliche, taschenbuchgroße Interface wird am RGB-Port des Amiga angeschlossen.

AG-6 (ca. 2800 Mark). Ein semiprofessionelles Genlock, der Firma Merkens für den Amiga 2000. Elf Ein- und Ausgänge, Spezialsignale wie Keyfill, Sync Out, RGB Out, Blanking, als auch die frei programmierbare Hardware- und Softwaresteuerung machen das AG-6 zum Kleinod.

IVS 1000 (ca. 4600 Mark). Das Spitzenmodell der Merkens Genlock Serie. Dieses für professionelle Anwendungen, in der Sendeabwicklung oder am Schnittplatz konstruierte Gerät läßt keine Wünsche offen. Es verfügt über die gleichen Features wie das AG-6 und besticht darüber hinaus durch exzellente Bildqualität. Farbauswahl, Farbumkehr, Farbumblendung in Schwarz oder Weiß sowie die stufenlose Ausblendung in eine dieser beiden Farben, sind nur einige Features dieses „deluxe"-Genlocks.

Magni 4000 Series, ein ProfiGenlock mit Sync- und Black-Burst-Ausgang für den Einsatz am elektronischen Schnittplatz. Ein- und Ausblendungen jeglicher Videoquellen können in verschiedenen Geschwindigkeiten vorgenommen werden und für Blue-Box-Effekte steht ein regelbarer „Chroma Key" zur Verfügung. Das Magni 4000 ist sowohl als NTSC-Version (zirka 1295 $) sowie als PAL-Version (zirka 1425 $) erhältlich.

Studio Genlock Lamm VCG-Serie (ca. 2000 Mark). Dieses für alle Videosysteme geeignete Genlock-Interface verfügt über eine Vielzahl professioneller Features zu einem erschwinglichen Preis. Blue-Box-Keying für den Amiga Hintergrund, LED-Anzeige für H-Sync und V-Sync sowie die VTR-Schaltung zur Bildstabilisierung sind die auffallendsten Merkmale dieses für die ganze Amiga-Reihe (A-500, A-1000, A-2000 A,B) geeigneten Geräts.

Broadcast Genlock Lamm VCG-Serie (ca. 3000 Mark). Die Profi-Ausführung der VCG-Serie für 1-Zoll-Recorder sowie Betacam und U-Matic Schnittplätze. Phasenkorrektur, Down-Stream-Keying (DSK), Black-Burst-Eingang sowie sämtliche Funktionen der Studioversion sind in diesem Nobel-Gerät enthalten.

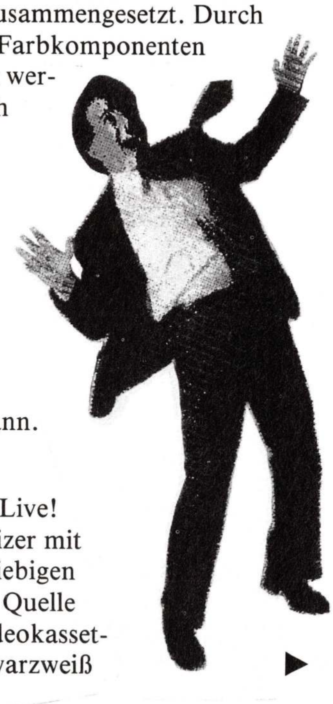
ProLock HV-1 (ca. 1200 Mark). Das preisgünstige Einsteigermodel von Michael Lamm Computersysteme. Speziell für den Einsatz auf VHS, Video 8 und Betamax ist dieses Modell geeignet. Im wesentlichen handelt es sich beim HV-1 um eine etwas abgespeckte Version des Studio Genlocks der VCG-Serie.

PBC PAL-RGB-Genlock (ca. 998 Mark), ein außergewöhliches Gerät welches nicht nur Videosignale sondern gleichzeitig auch Tonsignale verarbeitet. Das PBC Genlock verfügt über zwei getrennte Ausgänge für RGB- und FBAS-Signal.

DIGITIZER

DIGITIZER dienen dazu, stehende Bilder (zum Beispiel eine Lampe, ein Foto oder ein anderes greifbares Objekt) in zwei Dimensionen zu erfassen. Digitizer wandeln analoge Videoinformationen in digitale Informationen um. Der Computer entschlüsselt sie und verwendet sie zum Zeichnen der Pixel auf dem Bildschirm und für die Zuordnung der korrekten Luminanz sowie des jeweiligen Farbtons.

DIGI-VIEW (ca. 350 Mark) war eines der ersten Geräte für die Digitalisierung von Bildern mit dem Amiga. Es wird von der Firma NewTek hergestellt und angeboten. Digi-View arbeitet nach dem bekannten Dreiphasen-Verfahren (ähnlich wie auch Technicolor), bei dem ein Foto mit einer Schwarzweißkamera abgetastet wird. Es werden drei getrennte Aufnahmen (Rot, Grün und Blau) durchgeführt und zu einem Bild in 4096 Farben (oder mehr) zusammengesetzt. Durch diesen Trennprozeß der einzelnen Farbkomponenten kann eine hohe Bildqualität erzielt werden. Zur Zeit kann Digi-View noch keine Bilder im Overscan-Modus digitalisieren; NewTek hat jedoch diese Funktion für die nächste Update-Version in Aussicht gestellt. Ebenfalls in der Entwicklung befindet sich bei NewTek ein Gerät, mit dem ein Farbsignal von einer Kamera oder einem Videorecorder ohne Farbscheibe digitalisiert werden kann.



LIVE! (ca. 295 $). Die Firma A-Squared Distributions, Inc. hat Live! herausgebracht, einen Video-Digitizer mit der Fähigkeit, Einzelbilder aus beliebigen Videoquellen zu digitalisieren. Als Quelle ist eine Videokamera oder eine Videokassette möglich, die Bilder können schwarzweiß ▶

oder auch farbig sein. Live! kann ebenfalls eine Bildfolge mit 15 Bildern/sec. digitalisieren und anzeigen (der normale Wert für Video liegt bei 30 Bilder/sec.). Diese Bilder lassen sich im IFF-Format oder im RAM als komprimierte Bildfolge (benötigt bis zu 8 MByte) für eine Echtzeit-Wiedergabe speichern. Live! arbeitet im HAM- und im 32-Farben-Modus und besitzt zur einfacheren Installation eine Video-Eingangsbuchse sowie einen durchgeschliffenen Anschluß.

PERFECT VISION, ein von der Fa. SunRize Industries, Inc. entwickeltes Gerät, stellt eine Kombination von Digitizer und Frame Buffer dar. Es lassen sich Farbsignale aus einer beliebigen Quelle digitalisieren (mit 1 MByte RAM auf der Hauptplatine) und anzeigen oder als IFF-File zur Bearbeitung mit anderen Programmen speichern. Perfect Vision arbeitet mit allen Amigas und digitalisiert ein Bild in $\frac{1}{60}$ Sekunde. Es unterstützt die Auflösungsstufen 320 × 200 und 320 × 400 HAM mit jeweils 16 Farben.

FRAME CAPTURE der Fa. Mimetics Corporation ist ein Digitizer, der nur in Verbindung mit dem ReaSyn-Frame-Buffer dieser Firma zu verwenden ist. Er empfängt Standard-Farbsignale aus einer Videoquelle und digitalisiert diese für das Abspeichern im IFF-Format.

LAMM COLORSNAP-DIGITAL (Preis auf Anfrage), ein Gerät der Studio Serie von Michael Lamm Computersysteme. Alle Amiga-Auflösungen werden von diesem extrem schnell verarbeitenden (A/D-Wandler) Digitzer mit integriertem RGB-Splitter, erfaßt. Aller Voraussicht nach wird ColorSnap-Digital zum Jahresende 1988 lieferbar sein.
VD-3 (ca. 900 Mark), ein Echtzeit Digitizer der Firma Merkens. Die Erfassung eines Bildes erfolgt in 20 ms, und je nach Datenmenge können bis zu zehn Bilder pro Sekunde übertragen werden. Die Zeilenauflösung kann dabei 320 oder 640 Punkte betragen.

VD 2000 AMIGA (ca. 2500 Mark), das Spitzenmodell von Merkens Computer Video Systeme. Dieser, als Steckkarte in den Amiga 2000 erhältliche Farbdigitizer arbeitet in Realtime und mit 64 000 Farben. Das Gerät verfügt ferner über einen Video-Eingang sowie RGB-Ein- und Ausgänge.

FRAME-BUFFER

Dank der jüngsten Entwicklung auf dem Gebiet der digitalen Elektronik für Video-Anwendungen kann ein Frame Buffer bereits Video-Einzelbilder in bis zu 20 Millionen Farben aufnehmen und speichern. Aufgrund der eingeschränkten Hardware-Leistung ist der Amiga jedoch nur imstande, 4096 Farben am Bildschirm anzuzeigen. Die Informationskapazität ist jedoch für über zwei Millionen Farben ausreichend; leider kann die Hardware hier nicht mitziehen.

REASYN, der Frame-Buffer von Mimetics ermöglicht die Darstellung eines Bildes in 2 Millionen Farben, und zwar im Auflösungsmodus 740 × 480. Mit nur einem MByte RAM auf der Hauptplatine kann er an den Erweiterungssteckplätzen des Amiga 2000 angebracht werden. Mit dem Amiga 500 oder dem Amiga 1000 kann dieser Frame-Buffer nur konfiguriert werden, wenn eine externe Erweiterung für Amiga-2000-Karten verwendet wird. Das Board für die Einzelbildaufnahme wird hieran angesteckt, wodurch das Aufzeichnen von Video-Einzelbildern aus einer beliebigen Quelle sowie das Speichern auf Diskette oder Festplatte möglich ist. Zur Zeit arbeiten Byte by Byte und Aegis gemeinsam an einem Projekt, das die Anzeige von Grafiken aus Sculpt 3D, Animate 3D und VideoScape 3D mit dem ReaSyn Frame-Buffer erlauben soll.

VIDEO TOASTER (noch nicht erhältlich) ist die atemberaubendste Neuerung der letzten Monate. Das von NewTek entwickelte Gerät emuliert einen Zwei-Kanal-ADO (Ampex Digital Opticals, ein Studio-Gerät für die Erzeugung verschiedener Bildübergänge und Spezialeffekte). Es läßt sich an zwei Videoquellen (synchronisiert oder nicht) anschließen. Diverse Effekte lassen sich erzeugen, wie zum Beispiel ein sich drehender Würfel mit Bildern auf verschiedenen Flächen. Das alles in Echtzeitbetrieb, ohne Beeinträchtigung der normalen Amiga-Operationen. Es sind ebenfalls ein Digitizer, ein Frame-Buffer und ein Genlock eingebaut.

ENCODER

Als Encoder wird ein Gerät bezeichnet, das die einzelnen Farbkomponenten Rot, Grün und Blau mit den horizontalen und den vertikalen Synchronsignalen mischt, um ein Composite-Video-Signal zu erzeugen. Diese Geräte sind in zahlreichen Größen und Konfigurationen lieferbar, vom kleineren NTSC-Farbencoder auf Chip bis hin zum größten Modell für den Einbau in Video-Racks. Viele Fernsehstudios arbeiten mit diesem Gerät, um die Signale von Titelgeneratoren und Malsystemen in Composite-Video-Signale umzuwandeln. Leider kommt der hohe Preis Hobby-Filmern nicht gerade entgegen. Weiterhin läßt sich der Encoder bei PAL-Systemen einsetzen, wobei er die Standard-Komponenten Rot, Grün und Blau mit der gängigen Video-Software verwenden kann. So können verblüffende Effekte realisiert werden, ohne daß man das Programm wechseln muß.
Seit der Einführung des Amiga 1000 ist die Grundidee dieselbe geblieben: Der Amiga soll auf kostengünstige Weise Grafiken und Klang integrieren werden. Die für diese Zwekke notwendige Software und Peripherie ist nun endlich existent, zu Preisen, die auch vom Durchschnittsverbraucher verkraftet werden können. Zur Zeit wird an einer Vielzahl neuartiger Produkte entwickelt, und es ist zu erwarten, daß die Preise mit dem Heranreifen von Desktop Video weiter fallen.

(Wagland W. Strickland, mms)

Kooperiert Intel-Inboard mit A2000?

Ich besitze einen Amiga 2000 (A-Platine) und eine PC-Karte. Kann ich diese Karte mit dem Inboard 386/PC von Intel verbinden und zum Arbeiten bringen? Im PC10 (alt) arbeitet das Inboard angeblich. Darf man zwischen EGA-Karte und Amiga-Videoausgang mit einer Weiche auf einen Multisync hin- und herschalten?
H. Schneegold, Eibelstadt

Unseres Wissens hat bislang niemand versucht, die Amiga-PC-Karte mit dem Intel-Inboard zusammenarbeiten zu lassen. Im alten PC-10 arbeitet das Inboard tatsächlich. Ein Wechseln auf den Multisync ist nur mit dem Flickerfixer möglich.

Devices und Routinen

Betr.: D. Twigg-Flesners Antwort in AmigaWelt 3/88
Der Trackdisk-Device
Da nahezu alle bisherigen Amigas das gleiche Betriebssystem (Kickstart 1.2) besitzen, läßt sich aus der Aussage, das Trackdisk-Device befinde sich im Boot-ROM (mit der Begründung, daß sonst der A1000 keine Kickstart-Diskette lesen könnte), ableiten, daß der A2000 gar nicht funktionieren kann, denn schließlich verfügt dieser über gar kein Boot-ROM! Ich bitte daher alle Besitzer eines irrtümlich dennoch funktionierenden A2000, diesen zur Korrektur dieses unerwarteten Fehlverhaltens umgehendst zur Reparatur zu bringen. Sollte die Garantiezeit abgelaufen sein, so müssen Sie weiterhin mit einem fälschlicherweise funktionierenden Rechner vorlieb nehmen.
Das Trackdisk-Device befindet sich natürlich im Kickstart (-ROM), das Boot-ROM besitzt davon unabhängige Routinen zum Lesen der Kickstart-Diskette, die notwendigerweise dem Trackdisk-Device ähnlich sind.
Das FixHunk-Problem:
Nicht gekennzeichnete Hunk-Typen werden in MEMF_PUBLIC geladen. Dies ist im Moment jede Art von Speicher, vorzugsweise jedoch MEMF_FAST, sofern vorhanden. Unter zukünftiger Hardware (MMUs) wird sich die Bedeutung von MEMF_PUBLIC und diesem Hunk-Typ wandeln!
Der Zusammenhang mit 1.2 besteht nicht darin, daß die Programme an diese Kickstart-Version angepaßt werden müssen, sondern vielmehr darin, daß unter 1.2 autokonfigurierendes Fast-RAM nach dem RESET automatisch eingebunden wird, und mies programmierte Programme daher ebenso automatisch nicht funktionieren. Dies ist kein Fehler von 1.1 oder 1.2, sondern vielmehr der des Programmierers.

Ein kleiner Irrtum aus Sicht des Computer-Historikers in Dr. Kittels Antwort zum „FixHunk-Problem“. Die Unterscheidung zwischen Speichertypen ist in Kickstart 1.1 bereits durchaus möglich gewesen. Kickstart 1.0 kannte sie noch nicht und reagierte verständlicherweise mit dem Hinweis auf unzureichenden Speicher. Die übliche Entschuldigung, daß ein Programm für 1.1 geschrieben wurde, ist also sehr fadenscheinig.

Undokumentierte Systemroutinen:
Bei den in den FD-Files als „##private“ gekennzeichneten Routinen handelt es sich zwar um private Funktionen, jedoch erfüllen diese nicht immer die Funktion, die man aus dem Namen ableiten mag oder gar der offiziellen Dokumentation entnehmen kann (etwa DOS-Library).

Eine debug.library existiert nicht. Damit ist wahrscheinlich die „debug.lib“ gemeint, die von den vier genannten Exec-Funktionen Gebrauch macht. Es handelt sich um keine shared Library, sondern um eine linked Library. Weiterhin wird gerade die Funktion zur Formatierung von Zahlen auch vom console.device und von Exec selbst, etwa für die Gurumeldungen und nicht nur für den WACK, verwendet. Eine cstring.library ist zur Zeit nicht offiziell verfügbar, ebenso wie die clist.library, die bereits seit 1.2 aus dem ROM entfernt wurde. Die als „##public“ gekennzeichneten Funktionen der cstring.library entsprechen denen der beiden C-Compiler, sofern vorhanden.

Die info.library ist für die Workbench-Menü-Funktion „Info“ zuständig, RAMLIB dient dem automatischen Laden von diskresidenten Modulen (Libraries, Devices etc.) und deren Entfernung bei Speichermangel.

Eine Überarbeitung der bei Addison-Wesley erschienenen offiziellen Amiga-Dokumentation ist in Arbeit, ein Termin wurde aus verständlichen Gründen allerdings noch nicht genannt. Es würde mich aber sehr irritieren, wenn sie noch dieses Jahr erschiene.

Generell möchte ich von der Verwendung undokumentierter Funktionen DRINGENDST abraten. Der Amiga ist kein C64, Atari oder Apple (so interessant die letzteren auch waren)! Das Gleiche gilt für Programme, die etwa Datenstrukturen der IntuitionBase oder WorkbenchBase manipulieren. Alle derartigen Programme werden wahrscheinlich unter Kickstart 1.4, spätestens jedoch nächstes Jahr mit dem A3000, nicht einmal mehr die Oxidbeschichtung der Diskette wert sein.
Ralph Babel, Taunusstein

TAB bei Zweitlaufwerk

Mit Eurem TAB-Artikel, AmigaWelt 4/88, werden Besitzer eines Zweitlaufwerks Schwierigkeiten haben. Es fehlt die Anweisung zur Installation. Wer über ein zweites Laufwerk verfügt, kann sich nämlich das Kopieren der Befehle auf die RAM-Disk sparen und bei dem Befehl cd ram: beginnen, wobei dieser Befehl allerdings durch cd df<x>: tab zu ersetzen ist. Fishdisk 145 muß sich an diesem Zeitpunkt bereits im Laufwerk „x“ befinden.
A. Aigner, Bad Tölz

Rätsel um WB 1.3

Mit großem Interesse habe ich in der Ausgabe 4/88 die Dokumentation zur Workbench 1.3 gelesen. Ich klapperte einige Läden nach der Workbench ab, bekam aber immer die Antwort, daß es diese Version überhaupt nicht gäbe. Deshalb wende ich mich direkt an Euch. Für die Dokumentation muß Euch schließlich die WB 1.3 vorgelegen haben. Wäre es möglich, die Workbench bei Euch zu bestellen?
Chr. Schäfer, Alsfeld

Bei der von uns dokumentierten Workbench handelt es sich um die Entwicklerversion der WB 1.3 Gamma 7. Sie ist allerdings nicht im Handel. Bei Commodore herrscht bezüglich der Workbench 1.3 heillose Verwirrung. So gibt es bereits kommerzielle Software wie „Professional Page“ und „Comic Setter“, die bereits mit den Druckertreibern der Workbench 1.3 arbeitet. Wir fanden, daß auch die vorläufige Version eine Dokumentation wert sei, obwohl im Befehlssatz der entgültigen Workbench 1.3 sicher noch Veränderungen zu finden sind. Für alle, die dieselbe Idee wie Herr Schäfer haben: Bitte warten Sie die offizielle Version ab! Laut Commodore soll die neue Workbench Ende dieses Jahres in den Handel kommen. Man vermutet, daß noch einige Fehler behoben werden müssen.

Akustikkoppler für Amiga

Ich habe mir vor zwei Monaten einen Amiga 500 gekauft. Um auch mit anderen Usern in Kontakt zu kommen, möchte ich mir zusätzlich einen Akustikkoppler kaufen. Gibt es so etwas für den Amiga 500?
M. Erbe, Weinbach

PC-kompatible Akustikkoppler passen an den seriellen Port des Amigas. Akustikkoppler sind im Computer-Fachhandel erhältlich.

Laserdruck mit Hindernissen

Ein Laserdrucker ist ein teurer Spaß. Daß man damit aber nicht nur Spaß hat, zeigen Ihnen meine Erfahrungen. Ich habe mir vor kurzem für fast 10 000 Mark einen Laserdrucker gekauft. Dieses Hightech-Gerät aus Amerika oder Japan wurde in Einzelteilen geliefert. Also: einfach auspacken, anschließen, benutzen ging nicht! Nach dem Auspacken mußte ich das schwere Ding erst einmal aufschrauben, um Toner, Bildtrommel und Entwicklereinheit einzubauen. Ich als Nicht-Experte kam dabei ganz schön in's Schwitzen, denn die Bildtrommel darf dem Tageslicht höchstens 5 Minuten ausgesetzt werden, sonst sind 400 Mark beim Teufel. Immerhin, ich habe es geschafft. Laut Handbuch hätte ich nun lässig die Statustaste drücken können, und einen gestochen scharfen Probeausdruck bekommen. Leider kam nur zartgrau bedrucktes Papier aus dem Drucker. Ich befragte das Handbuch. Aha, man muß die Toner-Kassette herausnehmen und schütteln. Beim nächsten Druckversuch kamen endlich tiefschwarze Buchstaben. Das machte mich richtig übermütig und ich wollte den Laser unter Anleitung des Computers einen Text ausdrucken lassen. Da der Drucker ja angeblich Hewlett-Packard-kompatibel ist, hätten Steuerung und Schriftenwahl eigentlich kein Problem sein dürfen. Der HP-Treiber holte aber nur langweilige Standardschrift aus dem Drucker, von ▶

Schriftvariation keine Spur. Das Modell ist zwar in der Befehlssprache HP-kompatibel, kennt aber keine einzige Schrift des Vorbildes. Ich fand die Lösung: Der Drucker benutzt die Hewlett-Packard-Befehle, um die im Drucker fest eingebauten Schriften zu aktivieren. Was das nächste Mal beim Druck herauskam, war den HP-Schriften zwar ähnlich, aber eben nicht gleich. HP-Modelle haben nur zwei Schriften fest eingebaut; erst mit Zusatzmodulen hat man die versprochene Schriftenvielfalt. Da werden verschiedene Kassetten mit je einer Schriftart angeboten. Zu jeder Schrift gibt es einen speziellen Treiber. Ein Fachmann sagte mir: „Vergessen Sie alle Zusatzschriften, benutzen Sie nur die fest eingebauten." Ich fand das unglaublich, weil die Maschine doch extra einen großen Speicher für Zusatzschriften hat. Mein nächster Spezialist erklärte mir, daß es Softfonts gibt, Dateien, die vor dem ersten Ausdruck in den Drucker geschickt werden und ihm neue Schriftmuster beibringen. Dazu gäbe es Treiber für die üblichen Text- und DTP-Programme.
Mein nächstes Experiment: Ich probierte aus, ob der Drucker einen IBM-Grafikdrucker nachahmen kann. Das behauptet nämlich der Hersteller. Als ich nun die vom Handbuch vorgegebenen Befehle fix eintippte, arbeitete der Drucker wie ein IBM-Gerät, aber die HP-Emulation ging nicht mehr. Ich gab daraufhin die Befehle zum dauerhaften Einschalten der HP-Emulation (lt. Handbuch) ein. Ergebnis war ein spanischer Zeichensatz — ohne deutsche Sonderzeichen. Sprüche aus dem Handbuch wie „In diesem Emulationsmodus kann keine andere Schriftart eingestellt werden. Der Laserdrucker emuliert zwei weitere IBM-Schriften", haben mir die innere Organisation dieses Produktes nicht gerade erhellt. Gibt es nun eine oder mehrere Schriften?
Ein anderes Problem: Programmierfehler werden nicht analysiert. Es kommen keine Fehlermeldungen. Dann wollte ich eine Hardcopy anfertigen. Dazu habe ich die IBM-Graph-Emulation eingeschaltet. Sie enthält zwei Zeichensätze. Nur der zweite hat den gesamten IBM-Zeichensatz, der andere setzt an Stelle deutscher Sonderzeichen einfach Steuerbefehle. Ein Freund von mir löste das Problem mit der IBM-Escape-Sprache. Aber nicht, daß nun etwa alles funktionierte! Anstatt 80 Zeichen pro Zeile druckte der Laser nur 79, führte einen Wagenrücklauf durch und druckte dann das 80. Zeichen.
Ich fände es prima, wenn die Hersteller in Zukunft korrekte Handbücher und ein paar Druckertreiber auf Diskette mitliefern würden. Damit spreche ich sicherlich auch im Interesse vieler Leser.
R. Breitter,
Oldenburg

COBOL an der Uni

In Eurer Ausgabe 3/88 fragte Josef Weber nach einem COBOL-Compiler für den Amiga. Eure Antwort darauf war, COBOL sei eine „historische" Programmiersprache und nutze die vielfältigen Möglichkeiten des Amiga nicht aus. Ich finde diese Antwort unzutreffend. Wenn COBOL wirklich so historisch ist, wie Ihr behauptet, warum wird dann an so vielen Universitäten noch COBOL gelehrt (und programmiert)? Außerdem wurde — soviel ich weiß — das AmigaDOS in BCPL geschrieben, und ich behaupte, daß diese Sprache kein bißchen moderner ist als COBOL. Wahrscheinlich ist es relativ schwer, einen COBOL-Compiler zu schreiben, und niemand hat sich an diese Arbeit herangetraut. Ich finde, es ist eine Herausforderung für alle guten Programmierer, auf dem Amiga einen solchen Compiler zu schreiben. Und bestimmt gibt es genügend Amiga-Fans, die sich einen COBOL-Compiler genauso wünschen wie Josef Weber und ich. Meiner Meinung nach deutet dieses Interesse an der Sprache schon darauf hin, daß COBOL gar nicht sooooo veraltet sein kann, wie Ihr es darstellt.
Kertin Degen, Mainz-Gonsenheim

Echtzeituhr

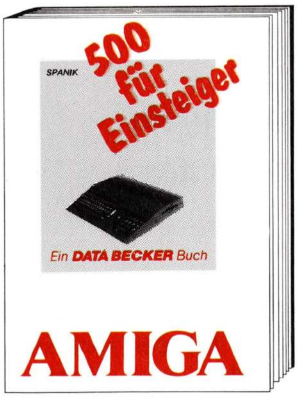
Betr.: Amiga 501 RAM/Clock Expansion
Mit der Speichererweiterung klappt es, aber nicht mit der Uhr! Ich habe den Computer bis zu 15 Stunden eingeschaltet gelassen. Trotzdem fiel die Uhr-Anzeige nach dem Ausschalten wieder auf die alten Werte zurück. Weshalb? Und wie funktioniert die Uhr richtig?
M. Estermann,
Oberengstringen/Schweiz

Mit der Erweiterung A501 verfügt der A501 über insgesamt zwei Uhren, entsprechend der Grundausstattung des A2000. Eine davon ist die batteriegepufferte Echtzeituhr der Erweiterungsplatine, die auch nach dem Ausschalten korrekt weiterläuft. Die andere wird mit Hilfe von Software und einem der beiden CIA-Bausteine realisiert. Letztere wird nach dem Einschalten oder Reset in Abhängigkeit des letzten Schreibzugriffs auf die Bootdiskette neu gesetzt. Der CLI-Befehl „Date" und auch das Uhrzeit- und Datumsfeld in „Preferences" beeinflussen diese letztere Software-Uhr und sind demnach in ihrer Wirkung nicht von Dauer.
Um den aktuellen Wert der Software-Uhr in die Echtzeituhr zu übertragen, muß das CLI-Kommando
SETCLOCK OPT SAVE
gegeben werden. Soll dahingegen die Software-Uhr, die ja auch vom Betriebssystem und allen Anwendungsprogrammen als Grundlage verwendet wird, auf den (hoffentlich) korrekten Wert der batteriegepufferten Uhr gesetzt werden, so geschieht dies durch
SETCLOCK OPT LOAD
Die Optionen „SAVE" und „LOAD" können mit dem Abspeichern und Laden der „flüchtigen" Systemzeit auf einen oder von einem dauerhaften Speicher verglichen werden. Das Speichern erfolgt üblicherweise einmalig pro Halbjahr (Sommerzeit) von Hand nach dem Setzen durch „Date" und das Laden automatisch bei jedem Start in der Startup-Sequence.
Unter der 1.3-Version von Preferences wird übrigens bei „SAVE" und „USE" automatisch auch die Echtzeituhr mitgesetzt. Der Fortschritt scheint sich nicht aufhalten zu lassen.
(Ralph Babel)

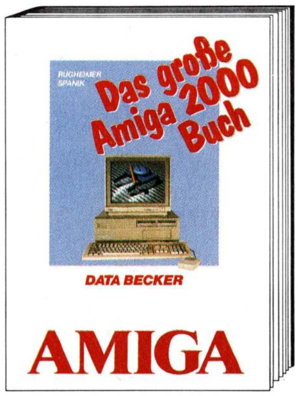
Ärger mit AmigaDOS-1.3

Die (in Ausgabe 4/88) beschriebene Vorversion Gamma 7 ist mittlerweile total überholt, einige Befehle sind wieder weggefallen, andere heißen anders, weitere sind dazugekommen und viele haben zusätzliche Funktionen bekommen. Mit anderen Worten: Die hier verbreiteten Informationen sind überholt und zum großen Teil falsch!
Es muß auch den Aussagen entgegengetreten werden, daß die Version 1.3 in Amerika schon frei beziehbar sei. Lediglich Anbieter von autobootfähigen Festplatten müssen mit Kickstart 1.3 fahren und brauchen dazu auch schon Teile der Workbench 1.3. Ansonsten befindet sich 1.3 zum gegenwärtigen Zeitpunkt, September 1988, noch im vertraulichen Vorstadium. Jeder, der auf einem mehr oder weniger dubiosen Weg an eine dieser Vorversionen gelangt ist, sei hiermit gewarnt: Diese Software ist noch nicht fertig, Fehler sind in jeder Hinsicht möglich, Inkompatibilitäten zur späteren endgültigen Version sind zu befürchten.
Dr. Peter Kittel
Commodore Frankfurt

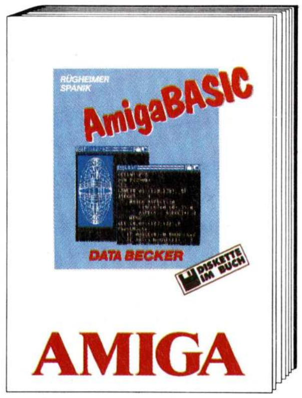
AMIGA BUCHHITS



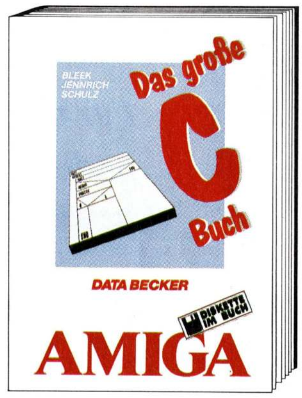
Gleich loslegen.
Bei „500 für Einsteiger" heißt es aufstellen, anschließen und sofort loslegen. Schnell und für jedermann verständlich zeigt Ihnen dieses Buch, was Sie mit Ihrem neuen Rechner so alles anstellen können. Workbench, AmigaBASIC, CLI und DOS – schon bald wissen Sie, worauf es ankommt.
Amiga 500 für Einsteiger
343 Seiten, DM 39,–



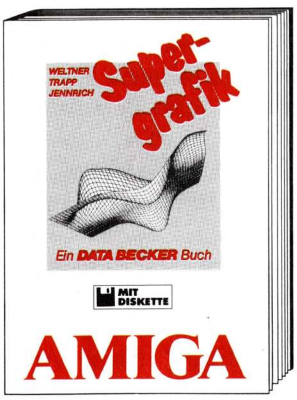
Ihr ständiger Begleiter.
Haben Sie einmal mit dem großen Amiga-2000-Buch gearbeitet, so wird es sicherlich seinen festen Platz neben Ihrem Amiga behalten. Denn dieses Buch bietet Ihnen mehr als eine detaillierte Einführung. Vom Laufwerkseinbau bis hin zum Kickstart im RAM wird hier echtes Profi-Wissen vermittelt.
Das große
Amiga-2000-Buch
Hardcover, 684 Seiten
DM 59,–



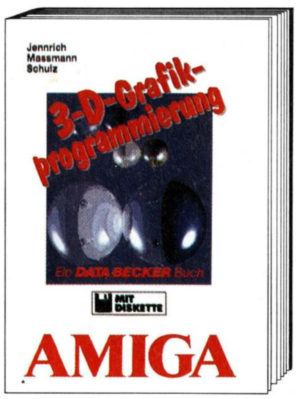
Amiga BASIC komplett.
Dieses Buch zeigt auf über 770 Seiten, worauf es beim Programmieren ankommt. Natürlich mit jeder Menge interessanter Programmbeispiele, die auch gleich auf Diskette mitgeliefert werden. Ebenfalls im Buch: eine detaillierte Beschreibung des AC-BASIC-Compilers.
AmigaBASIC
Hardcover, 775 Seiten
inkl. Diskette, DM 59,–



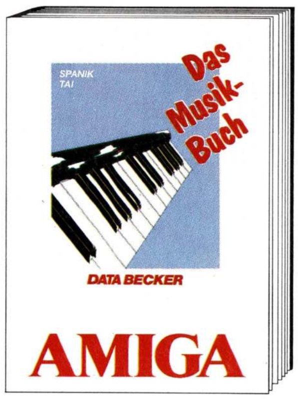
Profi-Programme in C.
Wenn Sie an C Spaß gefunden haben, gibt Ihnen das große C-Buch den letzten Schliff. Denn hier erfahren Sie nicht nur, wie ein C-Compiler arbeitet und wie Sie selbst die schwierigsten Probleme in C lösen, sondern auch, wie Sie eine optimale Benutzeroberfläche entwickeln.
Das große C-Buch
zum Amiga
Hardcover, 682 Seiten
inkl. Diskette, DM 69,–



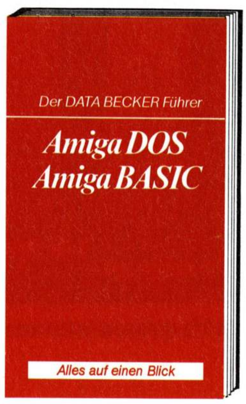
Das Buch zum Thema Nr. 1.
Grafik auf dem Amiga – mit dem entsprechenden Know-how ist hier fast alles möglich. Dieses Buch bietet es Ihnen: Nutzung der Libraries, die Register der Grafik-Chips, Aufbau und Programmierung von Screens, Windows, Halfbrite und Interlace aus BASIC und C u.v.a.m.
Amiga Supergrafik
Hardcover, 686 Seiten
inkl. Diskette, DM 59,–



Licht und Schatten.
Dieses Buch öffnet Ihnen den Weg in eine faszinierende Welt. Anhand ausgefeilter Grafikalgorithmen erzeugen Sie phantastische Bilder: naturgetreue Spiegelungen, Licht und Schatten. Alles absolut realistisch und automatisch berechnet. In allen Auflösungen mit bis zu 4096 Farben!
Amiga 3-D-Grafik-programmierung
Hardcover, 283 Seiten
inkl. Diskette, DM 59,–



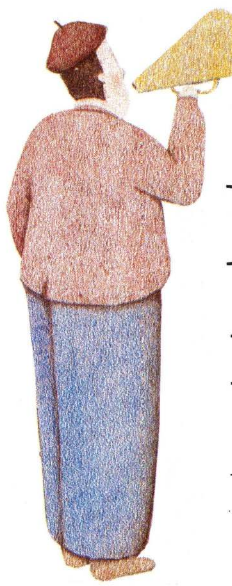
Da steckt Musik drin.
Hier werden Sie zu einem Komponisten ausgebildet, der nicht nur die notwendigen Grundbegriffe der Musiktheorie beherrscht, sondern auch modernste Technik einzusetzen weiß: Musikprogramme wie Sonix oder Audio Master, Sampler, MIDI-Interface und und und.
Amiga-Musikbuch
Hardcover, ca. 300 Seiten
DM 49,–
erscheint ca. 9/88



Der DATA BECKER Führer zu AmigaDOS & -BASIC
269 Seiten
DM 24,80

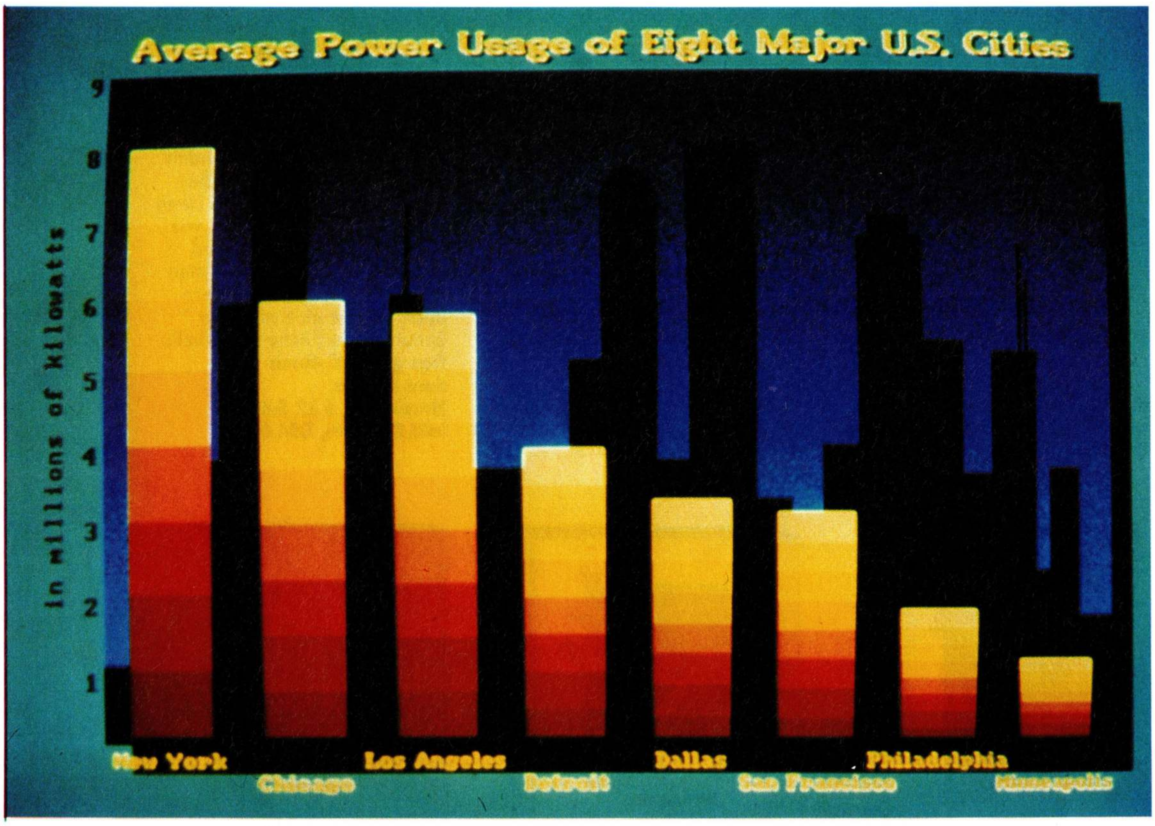
BESTELL-COUPON
Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir für meinen AMIGA
zzgl. DM 5,– Versandkosten
unabhängig von der bestellten Stückzahl
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei
Name
Straße
Ort

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010



Fernsehstudio in Deluxe-Qualität

Für ruckfreie Animationen, optisch einwandfreie Programmtafeln und Präsentationen der Extraklasse empfiehlt Electronic Arts das neueste Produkt aus der „Deluxe"-Palette.

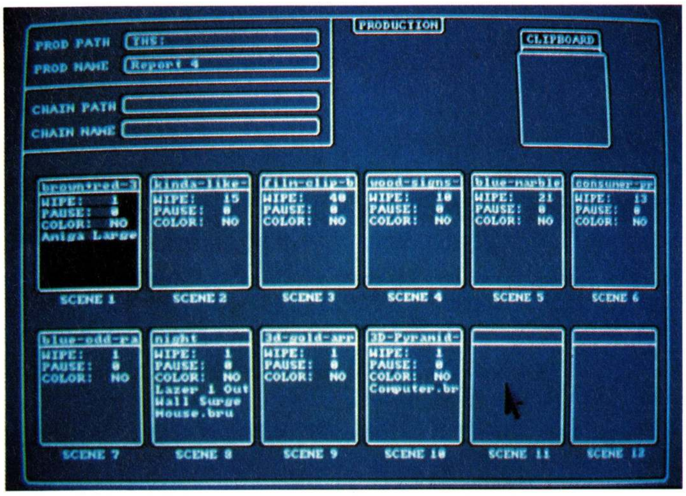


Bei Geschäftspräsentationen kann „Deluxe Production" seine Talente unter Beweis stellen.

Mit der wachsenden Bedeutung optisch aufbereiteter Information im Konsum- und Geschäftsbereich erlebte die Zunft der Grafiker und Werbetreibenden einen ungeheuren Aufschwung. Ebenso stiegen in den letzten Jahren die Kosten für animierte Werbespots, Ferngrafiken, Programmtafeln und geschäftliche Präsentationen. So ist es kein Wunder, daß die cleveren Programmierer Keith Mashavage und Chris Devine auf die Idee kamen, den Amiga mit solchen Aufgaben zu betrauen. Für „Electronic Arts" entwickelten sie „Deluxe Productions", ein Programm zur kostengünstigen und zugleich professionellen „Post-Production" (Vorproduktion).
Stark vereinfacht, läßt sich „Deluxe Production" als eine Art kleines Fernsehstudio beschreiben, welches ausschließlich der Produktion von Konserven (jederzeit abrufbare, vorgefertigte Produktionen) dient. Kamera, Rekorder und Mischpult werden dabei vom Computer simuliert. Um eine möglichst gute Qualität der bearbeiteten Produktionen zu erzielen, arbeitet „Deluxe Productions" nur in der höchsten Interlace-Auflösung. Für höchste Ansprüche bei der ruckfreien Animation von einzelnen Objekten wurde die Technik des Double Buffering gewählt. Der Programmspeicher ist dabei ständig mit zwei Bildern gefüllt. Während ein Grafikbild am Monitor sichtbar wird, baut sich dahinter das nächste Bild auf. Der einzige Nachteil dieses Prinzips ist der enorme Speicherplatzbedarf. Mindestens 1 Megabyte RAM sowie zwei Laufwerke sind nötig, um „Deluxe Production" zum Laufen zu bringen. Wer mit diesem Programm Ernsthaftes vorhat, wird sich sowieso mit einer Festplatte und zwei Megabyte RAM ausrüsten wollen. Ebenfalls empfehlenswert, sofern man das nötige Kleingeld besitzt, ist die Anschaffung eines „FlickerFixer" (siehe AmigaWelt 3/88) nebst Multisync-Monitor, der das ständige Arbeiten im Interlace-Modus erleichtert.

Disketten-TÜV inklusive

„Deluxe Productions" besticht durch einen hohen, professionellen Qualitätsstandard. So prüft das Programm die Disketten vor ihrem Einsatz auf Fehler. Unliebsame Überraschungen (Schreib-/Lesefehler) bei ge-



In einem Storyboard wird der Ablauf der Show genauestens festgelegt.

schäftlichen Präsentationen sind somit ausgeschlossen. Die im Programm enthaltenen Fonts entsprechen den Anforderungen der Fernsehanstalten. Die Fonts der Workbench können aufgrund ihrer geringen Höhe im HiRes-Modus nicht benutzt werden. Da den Fonts von „Deluxe Productions" jedoch die deutschen Umlaute fehlen, kann dieses Manko nur durch externe Zeichensätze (Zuma-, Cale-Fonts) ausgeglichen werden.

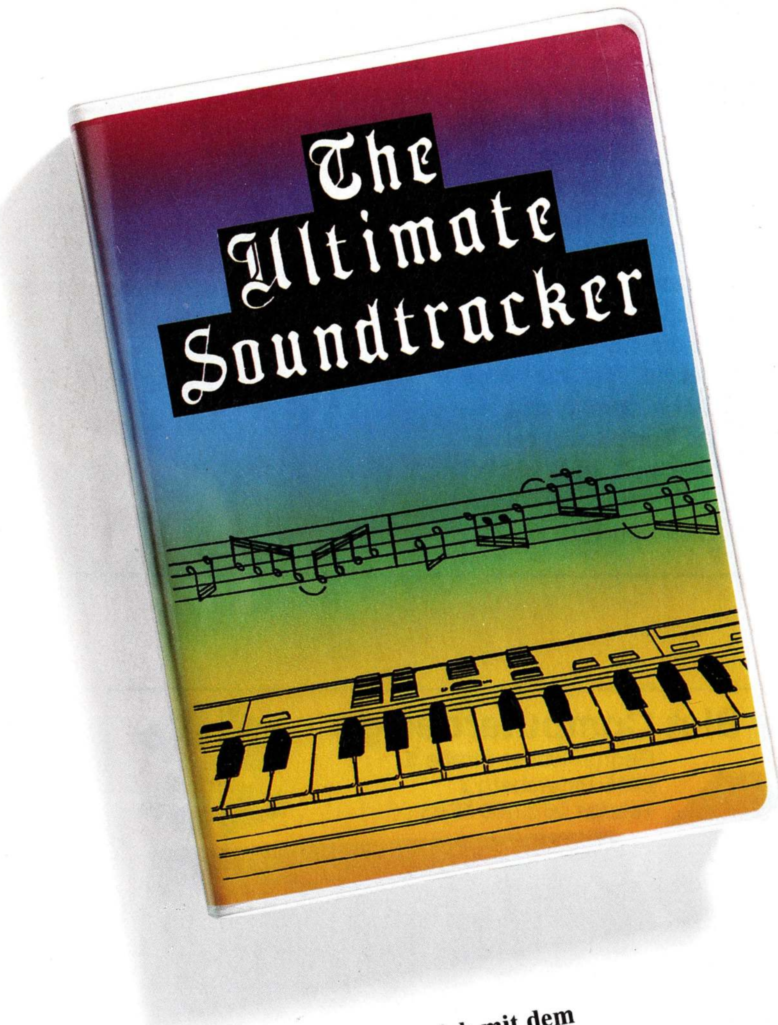
Nichts geht ohne DPaint

Für Besitzer einer Hard-Disk beinhaltet das Programm einen „Disk Installer". Verschiedene Produktionen können mittels „Chain Productions" verkettet werden. Für Präsentationen in Schaufenstern bietet sich die „Loop"-Funktion (Endlosschleife) an. Die Produktion eigener Spots und Animationen gestaltet sich aufgrund des logischen Programmaufbaus relativ einfach und leicht verständlich. In einem Storyboard werden sämtliche Daten der einzelnen Szenen festgehalten. Bis zu zwölf Szenen können für eine Produktion verwendet werden. Pro Szene kann man wiederum bis zu fünf Clips einbauen. Als Clip wird in diesem Fall ein animiertes Objekt bezeichnet. Aus einem Menü von mehr als 40 Wipes kann der jeweilige Übergang von Szene zu Szene gewählt werden. Sollte es Probleme mit den verschiedenen Farbpaletten einzelner Szenen geben, hilft „Full Color Assignment" die Übersicht zu bewahren. So gibt es eine Farbpalette für den Hintergrund und eine für die Objekte. Ferner kann man jede Farbe einer Palette in jeder gewünschten Reihenfolge mit „Color Cycling" zum Rotieren bringen. Ob es nun die Objekte sind, die mit DPaint erstellt und als Pinsel gespeichert werden, die Fonts oder die Bilder auf den Arts Disketten, nichts geht ohne Deluxe Paint II. Als Hintergrund können grundsätzlich alle Bilder eingesetzt werden, die das IFF-Format unterstützen. Ist die erste Produktion einmal fertig, so kann die Vorführung entweder „Nonstop", oder wie bei einer Dia-Show durch Mausklick von Bild zu Bild erfolgen. Mit dem frei kopierbaren „Player" kann man sein „Kunstwerk" dann in Umlauf bringen. Leider arbeitet „Deluxe Production" nicht in der PAL-Auflösung. Der Einsatz im Fernsehbereich wird dadurch eingeengt. Akustisch tut sich bei „Deluxe Productions" ebenfalls nichts.
(mms)

Info: Compustore
Fritz-Reuter-Str. 6,
6000 Frankfurt/Main 1,
Tel.: 0 69/56 73 99
Preis: 298 Mark

Soundtool für Spieleprogrammierer

Wer Spiele programmiert, möchte sein Meisterwerk mit einem perfekten Sound untermalen. Mit digitalisierten Klängen ist dies recht mühelos, aber nicht besonders originell. Karsten Obarski wollte für seine Programme etwas Pfiffigeres haben. Deshalb hat er ein Tool entwickelt, um eigene Musikstücke editieren und abspielen zu können.



Fantastische Songs lassen sich mit dem „Soundtracker" aus digitalisierten Samples editieren.

Im Auto-Spektakel „Ralley Master" fällt eine programmierte Musikunterlegung auf. Mit gesampelten Sounds hat Karsten Obarski fetzige Musik komponiert. Das Werkzeug dazu hat er selbst programmiert und über die Firma EAS Software auch anderen Programmierern zugänglich gemacht.

Der Lieferumfang des Programmes umfaßt neben zwei Disketten eine achtseitige Anleitung, die allerdings nicht alle Bedienungsgeheimnisse preisgibt. So wird erst nach einigem Herumprobieren im Programm verständlich, wie die 15 Soundplätze belegt sind und abgerufen werden können. Der Begriff Appregiato wird auch nach Studium diverser Musiklexika nicht verständlicher.

Lädt man die Hauptdiskette des Programms, wird man von einer recht simplen Titelmelodie begrüßt. Als Entschädigung wird im Vorspann auf zwei auf der Diskette vorliegende Stücke verwiesen. Eines davon ist der „Ralley Master Song". Diesen Song sollte man sich nach dem Laden des Soundtrackers anhören. Das Stück ist nicht nur musikalisch ein Hit, sondern ermöglicht einen guten ersten Überblick über die Funktionsweise des Programmes.

Der eigentliche Soundtrakker (auf der Diskette befindet sich zusätzlich noch ein Preset-Editor zum Bearbeiten der Samples) erweist sich als ausgesprochen durchdacht programmiertes Tool. Die wesentlichen Funktionen sind übersichtlich auf einer einzigen Arbeitsfläche dargestellt. Die 127 mitgelieferten Samples sind fast durchgängig von guter Qualität.

In der unteren Bildschirmhälfte des Programmes befindet sich der bei Soundprogrammierung wichtigste Arbeitsbereich: der Editor. Hier erfolgt die Eingabe der Notenwerte und die Verteilung der Töne auf einen der vier Kanäle mittels der Tastatur. Mit dem Editor kann ein jeweils 64 Sechzehntelnoten langes Pattern bearbeitet werden. Hierbei zeigt sich, daß bei der Programmierung vor allem auf Geschwindigkeit Wert gelegt wurde: Alle Bereiche des Pattern sind über die Cursortasten schnell zu erreichen. Nach dem Laden der Instrumente und der Noteneingabe kann das einzelne Pattern probeweise angehört werden, um eine Kontrolle über die Parametereinstellungen zu haben. Durch die Anzeige der gerade gespielten Notenwerte und die Möglichkeit, die Abspielgeschwindigkeit extrem weit zu senken, wird die Lokalisierung falscher Noten erleichtert. Die Pattern werden zu Songs zusammengefaßt und auf der Diskette gespeichert. Assemblerprogrammierer können den Song mit einer mitgelieferten Replay-Routine in ihre Programme einbinden. In der jetzigen Form ist das Programm nur Programmierprofis zu empfehlen. Vielleicht läßt sich Karsten Obarski dazu überreden, sein Können auch weniger spezialisierten Usern zur Verfügung zu stellen.

(Ralf-J. Triebel)

Info: EAS Software, Ferdinandstr. 16, 4630 Bochum, Preis: 99 Mark

AMIGOS

Hard Disk



... funktionelles Design und zuverlässige Technik

Die Festplatte wird anschlußfertig incl. Software mit durchgeführtem Expansionsport geliefert.

Die Festplatten arbeiten ohne Änderung mit Golem-Box, Kickstart-Modul A1000, Comspec-Box, Future-Vision, Profex zusammen. Die Festplatten sind für Amiga 500/1000 erhältlich (bei Bestellung bitte angeben).

Maße (in mm): H 65, B 330, T 330.
Erfragen Sie weitere Amigos-Produkte

Die zuverlässigen 20, 30, 40, 60 MB Festplatten gibt es zu Superpreisen:

SH20 1098,— DM (1 MB für 54,90 DM)
SH30 1298,— DM (1 MB für 43,27 DM)
SH40 1598,— DM (1 MB für 39,95 DM)
SH60 1898,— DM (1 MB für 31,64 DM)

FLESCH HÖRNEMANN
COMPUTER ELEKTRONIK

Schlägel-und-Eisen-Straße 46 · 4352 Herten · Telefon 0 23 66/5 51 76

Sprechblasenhelden erobern den Bildschirm

Comiczeichner können jetzt das Zeichenbrett wegwerfen. Mit dem „Comic Setter“ lassen sich phantastische Abenteuer bekannter und unbekannter Superhelden auf dem Bildschirm verwirklichen. Die Produktion eigener Hefte wird mit diesem Design-Studio zum Kinderspiel.

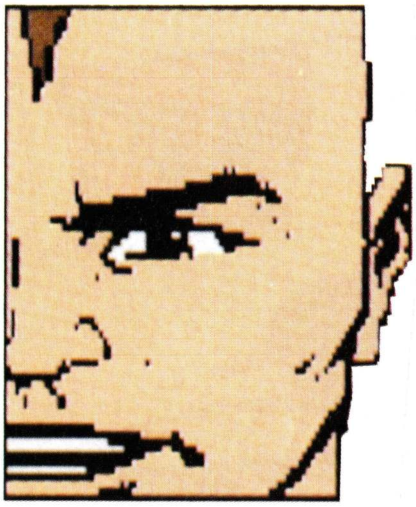


Jetzt ist der Comic Setter erhältlich. Drei Grafikmodule unterstützen den Bildromanautor bei der Arbeit.

Nach langer, sehr langer Ankündigungszeit ist der „Comic Setter“ des kanadischen Software-Herstellers Gold Disk endlich erhältlich. Es handelt sich bei diesem Programm um eine Mischung aus Mal- und Layout-Studio. Jedes Bild setzt sich aus einem Hintergrund (IFF-Bilder) und einem Vordergrund mit beliebig vielen Panels zusammen. Ist der Rahmen für den Hintergrund kleiner als das IFF-Bild, so kann man das Bild mittels vertikalem oder horizontalem Scroll-Balken dirigieren, um den gewünschten Bildausschnitt zu erhalten. Hintergründe können somit problemlos mehrfach verwendet werden. Wenn Captain Kirk und Mr. Spock sich vor einem Lift im Korridor des Raumschiffs unterhalten, kann die nächste Szene in einer Totalen den Gesamteindruck des Innenraums offenbaren. Jedesmal wird dabei dasselbe IFF-Bild als Hintergrund verwendet. Probleme bereitet normalerweise das Vergrößern von Bildschirmausschnitten. Die Zoom-Funktion bei Deluxe ▶



Selbst der Computer ist vor den eroberungswütigen Comichelden nicht mehr sicher, wie der Beispielausdruck einer Comic-Setter-Seite beweist.



Paint vergrößert nämlich in dem selben Maße die Pixel des Ausgangsbildes. Als Ergebnis erhält man so Bildeindrücke, die mehr an moderne Kunst, als an gelungene Grafik erinnern. „Comic Setter“ bedient sich daher eines besonderen Kniffes. Die Vordergrundfiguren werden als sogenannte Panels abgelegt. In ihrer Funktion sind sie mit den Brushes von DPaint zu vergleichen. Während die Brushes als kleine IFF-Screens abgelegt sind (jeder Bildpunkt wird auf einer fiktiven Speicher-Bitmap repräsentiert), legen die Panels nur die Daten von Anfangs- und Endpunkten gerader Linien und die Farben, mit denen die entstehenden Flächen zu füllen sind, ab. So wird in jedem Vergrößerungsmodus die jeweilig beste Auflösung und damit die geringste Treppchenstruktur erreicht. Diese Panels werden im Handbuch als „Polygone“ (Vielecke) beschrieben.

Wie bei der Produktion von Zeichentrickfilmen, bestehen die einzelnen Figuren aus vielen kleinen Einzelheiten. Auf der Clip-Art-Diskette befinden sich unzählige Files mit Gliedmaßen, Köpfen, Waffen und Lautwörter wie (Boing, Bum, Tschak). Jedes dieser Details ist als Panel abgespeichert. Nach dem Baukastenprinzip lassen sich aus diesen Panels Helden, Verbrecher, Monster und Roboter konstruieren. Ändert ein so zusammengesetztes Wesen den Gesichtsausdruck, so wird lediglich das Panel mit dem Gesicht ausgetauscht und mit dem „Polygon Tool“ in die richtige Größe gebracht. Wem dies zu anspruchslos sein sollte, der kann den Figuren mit dem eingebauten Grafikprogramm den letzten Schliff oder die persönliche Note geben. Besonders begnadete Künstler können selbstverständlich auch Charaktere mit einem anderen Malprogramm (Deluxe Paint II, Aegis Images) selbst produzieren.

Sind Hintergründe und Personen perfektioniert, widmet man sich dem Seitenaufbau. Schließlich soll auf einer komplett gefüllten Seite eine Geschichte erzählt werden. Sind diese Größenverhältnisse geklärt, werden die Sprechblasen eingefügt. Genügend Auswahlmöglichkeiten bietet ein Menü mit den verschiedensten „Word Balloons“. Auf einer formatierten Leerdiskette wird die vollendete Comic-Seite abspeichert und später per Farbdrucker auf echtes Papier gebracht. Dazu verwendet der „Comic Setter“ bereits die Druckertreiber der neuen Workbench 1.3.

In dem witzigen und ausführlichen Handbuch des „Comic Setters“ findet der User Hilfe bei allen anstehenden Problemen. Hier wird reich bebildert und gut verständlich jedes Werkzeug (Tool) vorgestellt. Die verschiedenen Pull-Down-Menüs werden ebenfalls erklärt und der Umgang mit Programmspezialitäten, wie „Bitmapped Graphics“, „Grids“ und „Balloons“ wird eingehend besprochen. Zum leichteren Einstieg in die Arbeit ist ein „Tutorial“ vorbereitet. Dies darf man sich jedoch nicht zu einfach vorstellen, denn der „Comic Setter“ ist ein sehr umfangreiches Programm. Neben der „Program Disk“ gibt es eine „General Clip Art“-Diskette. Insgesamt sechs „Comic Art“-Disketten kann der grafisch nicht so gewandte User zusätzlich erwerben. Die Schwerpunktthemen „Super Heroes“, „Funny Figures“ und „Science Fiction“, sind auf jeweils eine „Characters“- sowie eine „Background & Props“-Diskette verteilt. Obwohl es sich bei diesen Figuren nicht um „Orginal-Helden“ wie Batman, Captain Kirk oder Fix & Foxi handelt, so ähneln sie denen jedoch sehr. Die Comicfreunde unter den Amiga-Usern werden sich bestimmt darüber freuen, daß die Gold Disk bereits an einem „Movie Setter“ arbeiten, mit dem die gezeichneten Charaktere animiert werden können. Selbstverständlich wird dieses Programm, genau wie der „Comic Setter“, in einer PAL-Version erhältlich sein.

(mms)

Info: Gold Disk, 2171 Dunwin Drive, Unit 13, Mississauga, Ontario, Canada L5L 1X2; Preis: US$ 99, Comic-Art-Modul: US$ 35.

Bewegtes Comicleben

Movie Setter, das neueste Produkt von Gold Disk erlaubt nach nur kurzer Einarbeitungszeit farbenprächtige Animationen im Zeichentrickverfahren. Hintergründe können mit dem schon



integrierten Zeichenprogramm geschaffen werden. Jede beliebige IFF-Grafik läßt sich ebenfalls einbinden. Die Bildteile, die animiert werden sollen, lassen sich mit dem leicht zu bedienenden Objekt-Editor entwerfen und wie beim Zeichentrick in Einzelbildern Schritt für Schritt abfotografieren, also abspeichern. Zusätzlich bietet Movie Setter die Möglichkeit, die einzelnen Sequenzen mit IFF-Sounds zu versehen. Jetzt kann man das Ergebnis anschauen, wobei Movie Setter die einzelnen Bilder automatisch verbindet. Leichte Bedienbarkeit und saubere, ruckfreie Grafik sprechen für dieses ausgezeichnete Programm, das die Leistungsgrenzen des Amiga in Richtung Professionalität verschiebt.

Drei auf der Straße nach Modula

Wie ein Lastzug sollte sich eine Programmiersprache bei längeren Fahrten als zuverlässig erweisen und Kraftreserven besitzen, falls die Zeit drängt.

Bei Modula 2, noch von Niklaus Wirth begründet, handelt es sich um eine frisierte Version von Pascal. Das Modul-Konzept schaltet die meisten Ausfälle (wie automatisch variierbares Type-Casting), die nur mit einem Debugger behoben werden können, aus, und es zeigt sich auch in übersichtlicherer Form als die Libraries, da die Module einer Versionskontrolle unterliegen. Modula 2 geht mit Datentypen noch minuziöser um als C, doch es erlaubt als Verfahrensparameter offene Arrays (deren genaue Größe unbekannt ist) und behebt damit eine der größten Schwächen von Pascal. Weiterhin sind auch zwingende Daten- und Pointer-Typen möglich.

Alle drei Compiler für den Amiga, die Modula-2-Version 3.01a von TDI, die M2-Amiga-Version 3.1 und die Benchmark Modula-2-Version 1.03, verfügen über einen eigenen Editor, der in unterschiedlicher Form integriert ist, sowie über einen eigenen Linker. Alle drei unterstützen Module, die für die Amiga-Libraries und -Routinen definiert wurden, sowie weiterhin die Standard-Modula-2-Module, wie InOut und SYSTEM. Für alle drei ist eine Maschine mit 512 KByte ausreichend, obwohl ein Megabyte und mindestens zwei Diskettenlaufwerke, besser noch eine Hard-Disk, empfohlen werden. Unterschiede zeichnen sich hinsichtlich der Qualität von AmigaDOS und der Kernel-Interfaces, des erzeugten Codes und der Dokumentation ab.

TDI benötigt vier Anläufe

Der erste kommerzielle Modula-2-Compiler ist in drei Ausführungen lieferbar: Basic, Developers und Commercial. Abgesehen von den Unterschieden beim Preis bieten die Developers- und die Commercial-Version Source-Codes für diverse Module, sowie IFF-(Interchange File Format) und ILBM-(Inter-Leaved Bit Map) Unterstützung, einen Disassembler und eine Verweistabelle. Alle drei Ausführungen arbeiten auf der Grundlage des Original-Four-Pass-Wirth-Compiler-Modells, das zu einer gewissen Verlangsamung führt. TDI bietet verschiedene Compiler-Schalter für eine Code-Optimierung an, wie $OPT+ oder $OPT-, die jedoch in den Code integriert werden müssen. Diese Art von Optionen gehen auf die ersten CDC-Pascal-Compiler zurück. M2Amiga und Benchmark Modula-2 unterstützen ähnliche Compiler-Schalter, doch sind auch Befehlszeilenversionen lieferbar.

Der in den TDI-Compiler integrierte Editor soll dazu dienen, vom Compiler erzeugte Error-Files, also „erm.files“, zu finden, diese zu zählen und im Source-Code auf die Fehler hinzuweisen. Dies funktioniert allerdings nur in 50 Prozent der Fälle einwandfrei. Der Editor ist oftmals nicht imstande, die Fehler in dem vom Compiler erzeugten Error-File zu erkennen. Will man sichergehen, daß alle Bugs entdeckt wurden, ist man gezwungen, mit dem Editor den Code Zeile für Zeile durchzugehen. Bei den erm.files handelt es sich um Text-Files, die Zeile und Fehlernummer anzeigen. Mit anderen Editoren kann der Prozeß beschleunigt werden. Der TDI-Editor ist bei weitem nicht so leistungsfähig wie der MicroEmacs und kann lediglich dabei behilflich sein, Fehler aufzufinden, die man bereits identifiziert hat. Bei seriöseren Kompilationen läßt sich also mit dem symbolischen Debugger die Sezierung eines Totprogramms durchführen. Die Verwendung des Debuggers ist sicher wirkungsvoller als das willkürliche Verbreiten von WriteString-Calls im ganzen Programm.

Mit 163 KByte ist das Programm das größte unter den getesteten. Die Modul-Definitionen nehmen 90 Prozent des Platzes auf der Amiga-Diskette ein und machen es demnach unmöglich, den Compiler und Linker hinzuzufügen, wenn man eine getrennte Language- und Arbeitsdiskette anlegen möchte, wie dies mit M2Amiga und Benchmark möglich ist. Man muß entweder die Moduldiskette komprimieren, damit sie den Compiler aufnehmen kann, oder den Compiler, den Linker und den Editor auf die Arbeitsdiskette kopieren. Ist man im Besitz einer Harddisk, ist dieses Problem bereits gelöst.

TDI ist der umfangreichste und wohl auch der fehleranfälligste Compiler. Es kam beim Compilieren mit und ohne Fehler mehrmals zu Programmabstürzen und Guru-Meldungen, und auch bei dem erforderlichen AmigaDOS-Stack mit 30 KByte versagte TDI. Einmal hat er sogar eine Arbeitsdiskette ruiniert. Ebenfalls ärgerlich, doch weniger tragisch ist die Tatsache, daß sowohl der Compiler als auch der Linker stets den Fehler-Code 507 an AmigaDOS zurückschickten, auch wenn korrekt compiliert oder gebunden wurde. Sobald es den Error-Code empfängt, wird man von AmigaDOS aus dem Batch-Compiliervorgang geworfen.

Das Handbuch umfaßt Definitionen für jedes Modul sowie ein Index. Obwohl es klar gegliedert ist, kann man dem Handbuch manchmal nur schwer folgen. Im Kapitel Moduldefinitionen ist die Kopfzeile jeder Seite vom eigentlichen Thema, auf das sie sich bezieht, um eine Seite entfernt. Glücklicherweise sind die Kopfzeilen im Text und Index korrekt. Frustrierender ist allerdings, daß das M2-Konversionsmodul fehlt, das für die Number-to-String und die String-to-Number-Konversionen benutzt wird. Obwohl es im Handbuch beschrieben wird, ist es auf Diskette nicht enthalten.

Man sollte annehmen, daß der TDI-Compiler nach zwei Jahren auf dem Markt ausgereift ist. Seine Unzuverlässigkeit ist enttäuschend, zumal er bei den Benchmark-Zeiten gut abschnitt. Trotz des Four-Pass-Compilers lag er Kopf an Kopf mit dem M2Amiga.

Compilieren in einem Rutsch mit M2 Amiga

Von A. + L. Meier-Vogt aus der Schweiz wird M2 Amiga angeboten. Der Ein-Pass-Compiler (135 KByte) arbeitet auf Icon-Basis und läßt sich auf der ▶

Workbench ausführen. Dieser Compiler, der etwas schneller ist als der TDI, wies beim Compilieren und Binden in mehreren Läufen gleichbleibende Zeiten auf.

M2 Amiga ließ das Debugging von Benchmarks bei nicht übereinstimmenden und gemischten Typen. Der eingebaute Editor auf MicroEmacs-Basis erkannte die Error-Files des Compilers sofort; ein Fehlersuch-Modus, der mit dem des TDI vergleichbar ist, läßt sich über ein Pull-Down-Menü aktivieren. Im Gegensatz zu TDI wurden hier die Fehler stets gefunden, und darüber hinaus lassen sich mit ihm Compiler-Flags für das Ein- und Ausschalten von Fehlersuch-Bearbeitungszeit-Codes von der Compiler-Befehlszeile aus und auch vom Source-Code eines Files aus setzen. Insgesamt ließ sich der gesamte Editier-, Compilier- und Link-Prozeß wesentlich einfacher durchführen. Der Linker arbeitet schneller und gestattet sogar das Erzeugen von Icons für die fertigen Files.

Das Interfacing von Modula 2 und AmigaDOS ist mit M2Amiga ebenfalls wesentlich einfacher. Im Gegensatz zu TDI und Benchmark, bei denen gewisse Heap-Setup/Shutdown-Routinen aufzurufen sind, kann das M2Amiga-Bearbeitungszeitsystem das gesamte AmigaDOS-Setup- und Shutdown manipulieren. In Benchmark ist sogar zunächst die "math-Library" aufzurufen.

M2 Amiga zeigt sich in mancher Hinsicht etwas ungehobelt, so zum Beispiel bei der Dokumentation. Mit seinen dunkelblauen Buchstaben auf gräulichem Hintergrund ist das Handbuch nur schwer lesbar; das verwendete Papier ist äußerst dünn. Ein Index fehlt völlig, und die Modul-Beschreibungen sind nur schwer verständlich. Meier-Vogt hat einige Demos beigefügt, die einen Eindruck von der Leistungsfähigkeit des Programmes vermitteln sollen.

Man sollte auf eine gewisse Wartezeit gefaßt sein, wenn man den Compiler auf der Boot-Diskette installiert. Das Installationsprogramm braucht mehr als 14 Minuten, vielleicht deshalb, weil M2Amiga derart ausgelegt ist, daß es Systeme mit ein oder zwei Laufwerken handhaben kann und die gesamte Quelldiskette liest, bevor es mit dem Schreiben beginnt. Die Verzögerung ist notwendig, wenn der Compiler auf einer Diskette korrekt arbeiten soll. Wird das Programm nicht installiert, wird des öfteren nach der Systemdiskette gefragt.

Man ist ebenfalls gezwungen, eine Projektverzeichnis-Struktur aufzustellen, in der kompiliert wird und die für jedes Produkt die Unterverzeichnisse sym, obj und ref enthält. Allerdings erkennt der Linker diese Struktur nicht automatisch; er muß daher in den jeweiligen Objektfile geführt werden.

Sowohl der Compiler als auch der Linker erstellen Informationen für einen wahlweise lieferbaren Debugger, der im Handbuch erwähnt wird. Die einzige „Beschreibung" hierzu ist im Abschnitt über das Amiga-Bearbeitungszeit- Systemmodul zu finden. Der Abschnitt klingt eher wie ein Werbeslogan, da er auf die Existenz eines Debuggers hinweist, den man installieren kann; für weitere Hinweise wende man sich an den Händler. Soll der Käufer hier an der Nase herumgeführt werden?

Dritter im Bunde: BENCHMARK MODULA-2

Bei diesem von Avant Garde Software angebotenen Compiler handelt es sich um einen One-Pass-Compiler (103 KByte), der wahlweise über diverse Modul-Libraries verfügen kann. Auf der Hauptdiskette befindet sich ein Konfigurations-Tool, mit dem sich eine Vielzahl von Compiler-Optionen als Standard festlegen läßt, möchte man nicht mit den Standard-Optionen von Avant Garde arbeiten. Sie lassen sich von der Compiler-Befehlszeile oder vom Source-Code aus einstellen. Man hat die Möglichkeit, zwischen zwei integrierten Editoren zu wählen, die zudem den Vorteil besitzen, daß sich der Compiler und der Linker von diesen Editoren ausführen läßt. Ein willkommener Zusatz von Benchmark ist der Prozedur-Profiler für die Feinabstimmung einer Anwendung.

Das System läßt sich mit den wahlweise lieferbaren Modul-Libraries erweitern, zum Beispiel mit einer IFF-Modul-Library, einer „C- Language-Library" mit C-ähnlichen Funktionen, die auch Unbeugsamen den Übergang erleichtern sollen, und einer „Simple Amiga Interface"- Modul-Library mit Fenstern, Menüs und Gadgets zur einfacheren Handhabung.

Benchmark verwendet für seine Geschwindigkeit die Motorola Fast-Floating-Point-Library, doch wird der Variablentyp LONGREAL (64-Bit real) nicht unterstützt. Sowohl TDI als auch M2Amiga unterstützen diesen Typ; Benchmark muß hierauf über die IEEE Floating-Point-Math-Library zugreifen.

Die Variablen werden weiterhin durch den statischen Datenspeicherbereich eingeschränkt, der auf 32 KByte je Modul begrenzt ist. Dies ist nicht weiter tra-

Wegweiser nach Modula

Sich in einer fremden Sprache zurechtzufinden, ist selten leicht. Die folgende Liste soll eingeschworenen C-Programmierern das Umsteigen erleichtern.

```
The C STATEMENT                      IN MODULA-2 BECOMES:
if (x!= 15)                          IF (x # 15) THEN
{                                       WriteString('It is not 15! It ist ');
   printf(„It's not 15! It's %dn",x);   WriteInt(x);
}                                    ELSE
else                                    (* manipulate x here *)
{                                    END;
   /* manipulate x here */
}
```

Ähnliche Konvertierungen gibt es für while, do while, for und switch, wobei es in Modula-2 kein Äquivalent für GOTO gibt.

```
C                                     MODULA-2
#include..............................FROM module IMPORT xxx, xxx
#define...............................CONST
unsigned ..............................CARDINAL
unsigned long..........................LONGCARD
long int...............................LONGINT
int *fred..............................fred : POINTER TO INTEGER;
NULL...................................NIL
i++ ...................................INC(i);
i += 5.................................INC(i,5);
i——....................................DEC(i)   (* Ditto for DEC(i,5) *)
sizeof.................................SIZE (variable), TSIZE (type)
/* flags — several lines */............(* sets — several lines *)
   #define FLAG1  0x01                    CONST Flag1 = 1; Flag2 = 2;
   #define FLAG2  0x02                    Flag3 = OFH;
   #define FLAG3  0x0F                    TYPE FlagSet = SET OF [0..31];
   int flagword;                          flagword : FlagSet;
   flagword = FLAG1;                      flagword := flagword +
   flagword &= FLAG2;                     FlagSet {FLAG1};
   if (flagword & FLAG3)...               flagword := flagword —
   /* end of FLAGS */                     FlagSet {FLAG2};
                                          IF (Flag3 IN flagword) THEN ...
                                          (*END OF SETS*)
/* I/O */..............................(* I/O*)
   printf(„VARS %d %f %c %sn",         WriteString('Vars ');
   int_i, float_f, char_c,             WriteInt(int_i);
   string_s);                          WriteReal(float_f);
/* ditto for input */                  Write(char_c);
                                       WriteString(string_s)
                                       WriteLn;
                                       (*DITTO FOR INPUT*)
```

Benchmark Tests

Bearbeitungszeiten

	TDI	M2Amiga	Benchmark
Gesamt Test			
Compilieren	88x	55s	34s
Linken	149s	39s	42s
Größe des Exec.	33.8K	29.1K	42.4K
FFT			
Compilieren	56s	45s	27s
Linken	144s	32s	34s
Größe des Exec.	35.2K	19.6K	21.8K

Ausführungszeiten

	TDI	M2Amiga	Benchmark
FFT	29.3s	29.7s	28.7s
Permutation	5.9s	9.6s	5.3s
Towers	6.9s	9.3s	7.0s
8 Queens	2.8s	3.0s	2.7s
Int Mat Mult	12.5s	9.9s	8.6s
Real Mat Mult	16.5s	15.7s	21.2s
Puzzle	25.0s	28.6s	27.5s
Quicksort	6.2s	5.3s	5.5s
Bubble Sort	8.0s	8.7s	7.6s
Tree Sort	7.9s	14.2s	9.9s

gisch, wenn man die Variablen und Arrays zuvor aufspaltet, wobei man jedoch eventuell seine Daten neu ordnen muß, wenn man Programme zwischen Modula-2- Compilern porten möchte. Infolge der Anordnung der Datensegmente erzeugt Benchmark größere Code-Segmente als die anderen beiden Kandidaten.

Es ist erstaunlich, daß das Programm trotz des Range-Chekking und des Overflow-Chekking- Codes, den es besitzt, den Stack-Overflow nicht kontrolliert. Ich hatte das Limit in einem Fall überschritten und mußte von den Symptomen aus Rückschluß auf den Fehler ziehen.

Die Dokumentation ist vollständig und Demos sind reichlich vorhanden (Text, Grafiken, Sound etc.), obwohl die meisten aus Public- Domain-Programmen in C konvertiert wurden. Die Dokumentation ist umfassend, doch sie enthält leider keinen Index. Mit Hilfe des Abschnitts „Moduldefinitions-Verweistabelle" läßt sich ein Identifier nachsehen, das Modul auffinden, in dem es erscheint und dann seine Beschreibung im Buch nachschlagen. Sowohl das Handbuch als auch die bootbare Hauptdiskette enthalten Tutorials.

Auf der Wiegestation hilft kein Schummeln

Mit den Benchmark-Tests wurden die drei Programme auf die Waagschale gelegt und auf ihre Geschwindigkeiten geprüft, bei der Rekursion, beim Array-Index, bei Integer- und Real-Math, beim Pointer-Dereferencing und beim Zugriff auf die Struktur. Ausgeführt wurden ein schwer rekursiver Permutations-Calculator, ein Eight Queenssolver (50 mal ausgeführt), Puzzle (eine Math- und Array-trächtige Benchmark von Forest Baskett), Integer- und Real-Matrix-Multiplikatoren (40 × 40 Arrays), drei Sortierungen (Quicksort, Tree Sort und Bubble Sort, je mit 5000-Element-Arrays-Zufallszahlen). Weiterhin wurde ein Fast-Fourier-Transform-Programm mit einem Array von 256 komplexen Zahlen getestet, die als Records beschrieben waren.

Ursprünglich war dieser Test mit den anderen gekoppelt, doch die kombiniert statische Datengröße war zu umfassend für Benchmark Modula 2. Alle Benchmarks geben nur an den Bildschirm aus, denn das Input/Output ist eher die Aufgabe eines Betriebssystems als die eines Compilers.

Als Testgrundlage wurde ein System mit zwei Laufwerken verwendet; eingesetzt wurden Compiler, Linker und Modul-Beschreibungen zur Diskette sowie die AmigaDOS-Befehle auf der RAM-Disk. Soweit möglich, verwendete ich den System-Editor des jeweiligen Programms; bei TDI mußte ich jedoch des öfteren auf Micro Emacs zurückgreifen. Jeder Prozeß wurde mit DateStamp AmigaDOS getrennt getimed, um jeweils die Zeit vor und nach dem Aufrufen des Moduls zu messen. Beim Wechsel des Compilers wurde lediglich der Name der Datentypen geändert, die von den Amiga-Modulen importiert wurden, sowie die Namen der importierten Module selbst. Der kombinierte Test wurde mit 725 Zeilen ausgeführt, das FFT mit 260 Zeilen. Die Unterschiede bei den Importen, beim Type-Casting sowie den Setup-Calls ließen die Zeiten der Compiler unterschiedlich ausfallen.

Am Ende steht der Inspektionsbericht

Alle drei Kandidaten weisen ähnliche Geschwindigkeiten bei der Code-Erzeugung auf, wobei der Gleitkomma-Code von Benchmark leicht in Führung liegt. Der TDI-Compiler ist nicht zu empfehlen, zumindest nicht, bevor der Herausgeber die Probleme der Version 3.01 a behoben hat. Der Code besitzt zwar die gleiche Qualität wie der des M2Amiga-Compilers, doch verliert man durch das ständige Debugging und den Neuaufbau nach den Guru-Meldungen äußerst viel Zeit. Die anderen Tools sind auch nicht gerade überwältigend. Da zeigt sich M2Amiga schon etwas solider, doch läßt die Dokumentation zu wünschen übrig, die gerade für den Anfänger keine große Hilfe darstellt. Der Compiler von Benchmark, der noch sehr neu ist, muß sich zunächst noch durchsetzen. Trotz seiner Schwerfälligkeit an kritischen Stellen ist der Benchmark-Code empfehlenswert, der übrigens der umfassendste ist. Er schneidet zumindest genausogut ab wie der M2Amiga-Compiler, und man bekommt einfach mehr Leistung für sein Geld. Das im nächsten Jahr erscheinende Modell soll eine noch bessere Leistung erbringen.

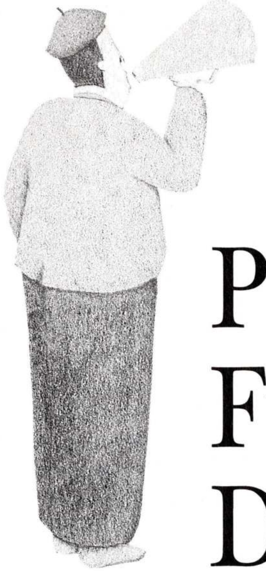
(David T. McClellan)

Adressen

M2-Amiga
A + L Meier Vogt,
Im Späten 23,
CH-8906 Bonstetten/ZH,
Tel.: 00 41/1/7 00 30 37

Benchmark, Modula-2
Avant Garde Software
2213 Woodburn,
Plano, TX 75075, USA,
Tel.: 0 01/2 14/9 64 02 60

TDI Modula-2
TDI Software Inc.
10355 Brockwood Rd.,
Dallas, TX 75238, USA,
Tel. 0 01/2 14/3 40 49 42



Public-Domain-Software-Fundgrube für Desktop-Video-Anwender

Desktop Video ist ein besonders arbeitsintensives Anwendungsgebiet — egal ob es sich um Computerfilme, um Titeltexte oder um das Mischen von realen Filmen mit Trickfilmen und Computergrafiken handelt. Public-Domain-Programme bieten hier ihre Unterstützung an.

Für Amiga gibt es mittlerweile zahlreiche kommerzielle Programme, die den Desktop-Video-Anwender bei seiner Arbeit unterstützen. Obwohl derartige „Komplettlösungen“ in der PD bislang noch nicht zu finden sind, lohnt sich das Durchforsten der frei kopierbaren Programme. Auch in Sachen Desktop Video erweist sich Public Domain als wahre Schatztruhe.

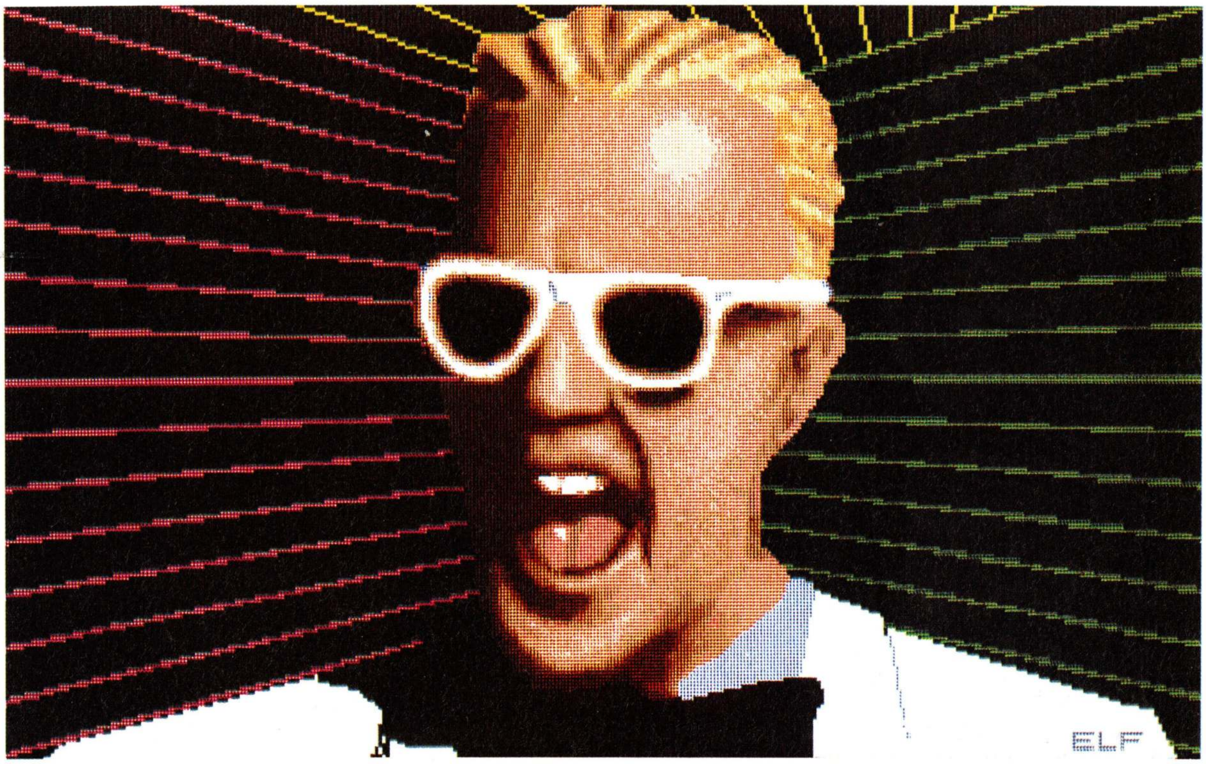
Rohstoffe ohne Grenzen

Auch mit den besten (kommerziellen) Programmen lassen sich anspruchsvolle Grafiken oder Animationen nur mit großem Zeitaufwand realisieren. Alternativ dazu kann man auch auf den großen PD-Vorrat an Grafik- und Slideshowdisketten zurückgreifen. Hier findet man für viele Zwecke thematisch passendes Material, das übernommen oder nach eigenen Vorstellungen weiterbearbeitet bzw. verfremdet werden kann. Bei der Herstellung von Videoproduktionen für kommerzielle Zwecke ist es jedoch ratsam, sich über eventuell bestehende Urheberrechte sachkundig zu machen. Witzige Animationen, die sich gut dazu eignen, einen Film aufzulockern, finden sich unter anderem auf zahlreichen Fishdisks, ebenso verschiedene Uhrenprogramme, die für Zeiteinblendungen in Realfilme herangezogen werden können. Besonders ergiebig für Aufgaben im Bereich der Vertitelung sind die Public-Domain-Fonts, die dem Anwender eine größere Variationsbreite verschaffen.
Da die Suche nach diesen Materialien oft der nach der berühmten Nadel im Heuhaufen gleicht, haben wir — ohne Anspruch auf Vollständigkeit erheben zu wollen — einige Fundquellen in Tabellenform zusammengestellt. Weitere Hilfestellung geben die Katalogdisketten der PD-Anbieter oder die systematischen Programmlisten der PD-Handbücher von technicSupport. Empfehlenswert sind auch Spezialreihen wie die RMS-Reihe, die ausschließlich Bilder enthält, wobei sich jede Diskette auf eine bestimmte Thematik konzentriert. Alle Disketten sind randvoll, dennoch differiert die Zahl der auf ihnen enthaltenen Bilder abhängig von Auflösung und Platzbedarf. Digitalisierte Bilder von sehr hoher Qualität finden sich ebenso wie mit Malprogrammen erstellte Kunstwerke, die übrigens sehr oft mit sehenswerten Color-Cycling-Effekten ausgestattet sind.

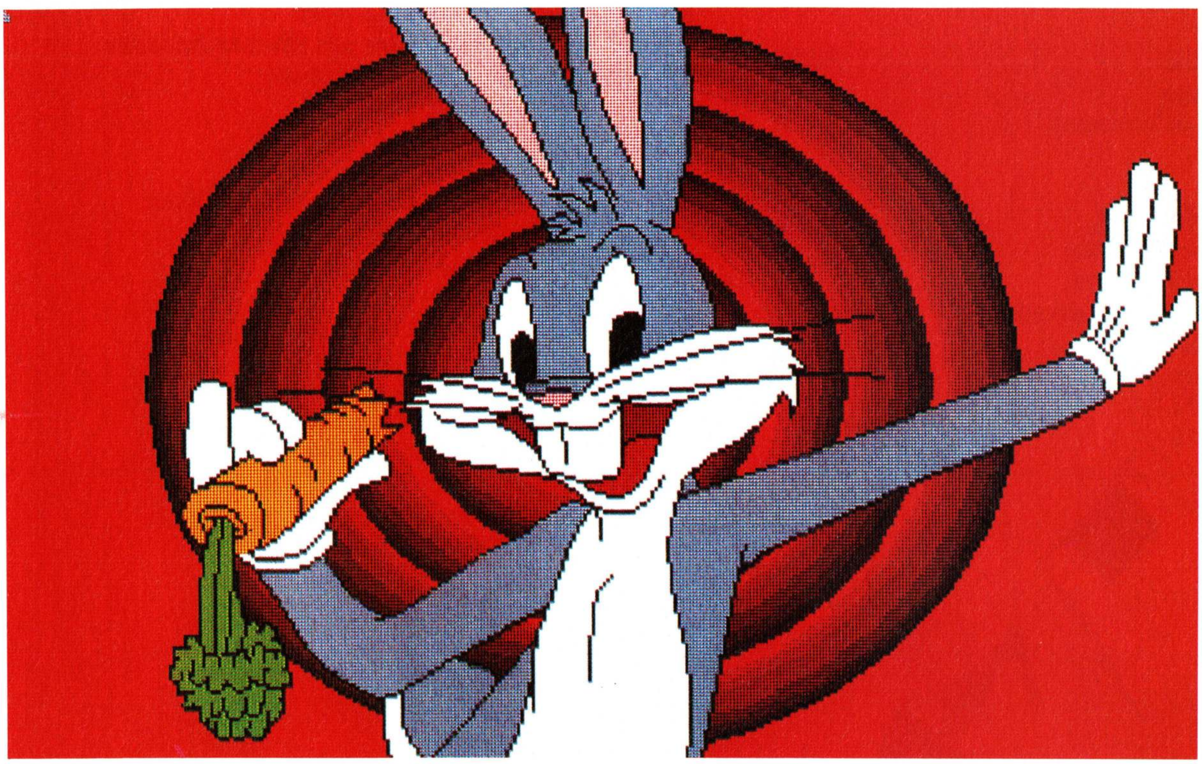
Grafikverarbeitung mit PD-Programmen

Auf der anderen Seite bietet die Public Domain auch viele Programme, die die Erstellung von Grafiken und ihre Weiterverarbeitung unterstützen. So findet man auf Fishdisk 52 mit Vdraw 1.19 ein durchaus brauchbares Malprogramm, auf Fishdisk 71 ein Programm namens FPic zur Nachbearbeitung von SW-Grafiken. Ray-Tracing-Programe findet man auf Fishdisk 95 (A-Render), Fishdisk 146 (C-Light) oder RPD 021 (DBW-Render). Erfordert beispielsweise ein Lehrfilm die Einblendung technischer Zeichnungen, dann lassen sich diese mit dem Programm mCAD von RPD 51 erstellen. Bei genügendem Speicherplatz läßt sich mit Hilfe von QuickFlix (Fishdisk 106) auch eine stufenweise Entstehung solcher Zeichnungen verwirklichen. Mit QuickFlix lassen sich Sequenzen von IFF-Bildern abspielen und zu kürzeren Trickfilmen verbinden. Auch für die Vertitelung ist das Programm geeignet. Das „Lasers“-Beispiel auf der Fishdisk zeigt, wie unter geschickter Ausnutzung des Color-Cyclings originelle animierte Schrifteinblendungen realisiert werden können. Jedoch lassen sich auch ganz andere Programme nutzen. Warum nicht einmal ein Titelbild mit dem

Fortsetzung auf Seite 54 ▶



„Max Headroom“ fehlt in der RMS-Sammlung . . .



. . . ebensowenig wie „Bunny der Hase“

Animationen & Trickfilme

Name	Diskette	Kurzbeschreibung
Juggler	Fish 057	Der berühmte Jongleur
BeachBirds	Fish 112	Vögel fliegen über Steilküste
Focus	Fish 112	Wehrwolfgesicht wird erzeugt
RGB	Fish 112	Verformung eines Gesichts
Killer	Fish 115	Amiga gegen Atari und Mac
Kahnankas	Fish 116	Das Managerspiel
Rocker	Fish 116	Schaukelstuhl in Bewegung
F-15	Fish 116	3D-Flugzeugmodell
Stars	Fish 118	Raumschiff im Weltraum
Car	Fish 123	Auto trifft Einrad
ElGato	Fish 125	Laufende Katze
HBHill	Fish 126	Steppender Fred Astaire
OnlyAmiga	Fish 126	Pyramiden jonglieren Bälle
Ripples	Fish 127	Betrachter kreist um Amiga-Flagge
Nemesis	Fish 127	Weltraum-Computerfilm
Berserk	Fish 132	3 Einräder tanzen auf Ball
BoingThrows	Fish 134	Korbwurf eines Basketballers
Spinner2	RPD 059	Rotierende Kugeln
Spiral	RPD 070	Zeichnet farbige Spiralen
2Balls	RPD 079	Zwei verspiegelte Kugeln
DemoReel	RPD 086a/b	Bilder & Animationen von NewTek
Henry.anim	RPD 106a	Laufendes IC
Lor.movie	RPD 106a	3D-Drahtmodell, animiert
Boing3D	RPD 110	Hüpfender Ball
Guitar	RPD 110	Noten kreisen um Gitarre
Jet.movie	RPD 110	Jet fliegt aus dem Monitor
Weird_Bird	RPD 110	Ray-Tracing-Vogelmodell

Die RMS-Reihe

Nr.	Thematik	Bildanzahl
1	Fahrzeuge	22
2	Fliegen & Weltraum	19
3	Comics	16
4	Tiere	19
5	Labels & Logos	26
6	Ray-Tracing-Bilder	10
7	Technische Geräte	24
8	Color-Cycling-Animationen	15
9	Köpfe & Figuren	21
10	Gesichter	15
11	Erotik *)	15
12	Erotik *)	16
13	Karikatur & Malerei	15
14	Malerei & Impressionisten	20
15	Bilder von P. Nagel	18
16	Bilder von Escher	13
17	Heavy Metal	15
18	HAM-Bilder	16
19	SW-HiRes-Bilder	8
20	Farb-HiRes-Bilder	7
21	HiRes-Frauenbilder	7
22	Romantik & Landschaften	16
23	Magie & Sage	18
24	Erotik *)	11
25	Digitalisierte Tiere	13
26	Science Fiction	11
27	Fantasy	9
28	NASA & Weltraum	12

*) Nur gegen Altersnachweis (über 18 Jahre) lieferbar.



PD-Bildmaterial

Diskette	Inhalt
Fish 030	Verschiedene Bilder
Fish 020	Mandelbrot-Grafiken
Fish 039	Ray-Tracing-Bilder
Fish 044	Ray-Tracing-Bilder
Fish 045	Verschiedene Bilder
RPD 003	Verschiedene Bilder
RPD 005	Defender of the Crown
RPD 009	Verschiedene Bilder
RPD 012	Gemälde, digitalisiert
RPD 020	Digitalisierte Bilder
RPD 027	Nagel-Bilder und SW-Landschaften
RPD 031	Tierbilder, digitalisiert
RPD 039	Science Fiction
RPD 050	Weltraum und Fantasy
RPD 052	Verschiedene Bilder
RPD 055	Mathematisch berechnete Bilder
RPD 061	Verschiedene Bilder
RPD 068	Verschiedene Bilder
RPD 078	Fantasy
RPD 082	Frauenbilder, digitalisiert
RPD 091	Ray-Tracing-Bilder
RPD 096	Verschiedene Bilder
RPD 099	Tierbilder, digitalisiert
RPD 105	Verschiedene Bilder
RPD 111	Geometrische Objekte, 3-D
RPD 114	Bilder von P. Nagel
RPD 119	Verschiedene Bilder

Disketten mit Zeichensätzen

Fish 034
Fish 061
Fish 071
Fish 081
Fish 105
Fish 135
RPD 008
RPD 063
RPD 115

Info: RMS; C,S.S.;
Auf der Warte 46,
6367 Karben 1,
Telefon: 06039/5776
Fish, RPD: div. Anbieter

Fortsetzung von Seite 50

Gag-Programm Melt-ILBM von RPD 026 wie Wachs zerfließen lassen, bevor es wieder ausgeblendet wird? Der Phantasie lassen sich keine Grenzen setzen.

Texteinblendungen mit SportsText

Während Filmvorspanne gewöhnlich mit aufwendigen Grafiken und Effekten ausgestattet werden, genügen für rein informative Texteinblendungen während des laufenden Filmes weniger anspruchsvolle Verfahren. Effekthascherei ist hier nicht gefragt, da sie in den meisten Fällen überzogen wirken und den Betrachter zu sehr ablenken würden. Genau für diese Art sachlich-informativer Einblendungen wurde das Programm SportsText konzipiert, das man auf der RPD-Diskette 108 findet. Wie der Name andeutet, hatte der Autor in erster Linie die Vertitelung von Sportdokumentationen im Sinn: die Einblendung eines Spielstandes, eines Torschützen usw. Obwohl es sehr einfach gehalten ist, kann das Programm diese Aufgaben recht gut bewältigen. Die Version 1.01 arbeitet in Lo-Res-Auflösung und kann jeweils eine Textzeile einblenden. Nach dem Anklicken des SportsText-Icons öffnet das Programm einen eigenen Screen, über dessen Menüleiste sich alle Einstellungen vornehmen lassen, wozu man lediglich Schalter oder Schieberegler anklicken muß. Alle Menüpunkte lassen sich auch über Tastenkombinationen erreichen, die in der Menüleiste selbst aufgeführt sind. Nach dem Start ist auf dem Bildschirm eine Textzeile sichtbar, an der alle im Settings-Menü vorhandenen Einstellmöglichkeiten getestet werden können.

Im Select Fonts-Untermenü stehen acht verschiedene Fonts (Topaz 8, 9, 11; Ruby 10; Diamond 12, 20; Sapphire 14 sowie Garnet 16) zur Verfügung, die leider nicht ohne weiteres gegen andere Fonts ausgetauscht werden können: ein Problem, das wir weiter unten in Angriff nehmen werden. Soll die Schrift mit einem Schatten versehen werden, so kann dieser im Select Dropshadow-Untermenü in drei verschiedenen Stärken und 4 Positionen gewählt werden. Wahlweise kann der Text auch mit einem farbigen Balken unterlegt werden. Die Farben von Schrift und Schatten werden im Select Colors-Menü festgelegt, die vertikale Position der Textzeile nach Anwahl von Select Position mit der Maus bestimmt. Leider wird der Text immer mittenzentriert ausgegeben. Das kann nachteilig sein, wenn z.B. ein Spieler aus mehreren hervorgehoben werden und bezeichnet werden soll. Mit etwas Geschick läßt sich dieses Problem aber bei der Eingabe des Textes aus der Welt schaffen. Nicht beseitigen läßt sich der Umstand, daß das Programm den PAL-Bildschirm nicht erkennt. Texte lassen sich daher nicht auf die unteren 56 Bildschirmzeilen positionieren.

Alle so gewählten Einstellungen lassen sich im Project-Menü mit Save Settings abspeichern und mit Load Settings wieder zurückladen. Ein weiterer Menüpunkt (Get The Script) dient zum Einladen der zuvor erstellten Texte. Der Dateiname läßt sich, ebenso wie bei Load Settings und Save Settings nicht frei wählen.

Die gewünschten Texte müssen mit Hilfe eines Editors wie dem ED in einer speziellen Datei abgelegt werden, die den Namen „SportsText.script" tragen und in demselben Verzeichnis abgespeichert werden muß, in dem das Programm selbst liegt. Jede Zeile dieser Datei entspricht genau einer einzublendenden Zeile. Um die unterschiedlichen Titel gezielt abrufen zu können, ist dem eigentlichen Text eine zweistellige Kennung (dabei sind Ziffern und Buchstaben erlaubt), gefolgt von einem Leerzeichen oder einem Doppelpunkt zur Trennung vom Text, voranzustellen. Eine typische Datei sieht dann wie folgt aus:

a1: 57. Min: 3:1 Peter Peters
bb: Matchpoint 21: Weite: 6,37 m

Beim Abspeichern ist darauf zu achten, daß eine schon bestehende Datei namens SportsText.script nicht überschrieben wird. Besser ist es, eine solche Datei schon vor dem Aufruf des Editors mit dem Rename-Befehl umzubenennen.

Nach dem Aufruf von SportsText ist die Textdatei über den Menüpunkt Get The Script einzulesen, gegebenenfalls sind auch noch die Settings zu laden. Über die Tastatur wird nun blind die Kennung der gewünschten Textzeile (z.B. a1) eingegeben. Ist der Film an der gewünschten Stelle angelangt, genügt ein Druck auf die Return-Taste, und die gewählte Textzeile erscheint. Mit der Leertaste läßt sie sich wieder ausblenden. Fehlerhafte Eingaben werden ebenfalls mit der Leertaste storniert, dabei verschwindet allerdings die aktuelle Einblendung.

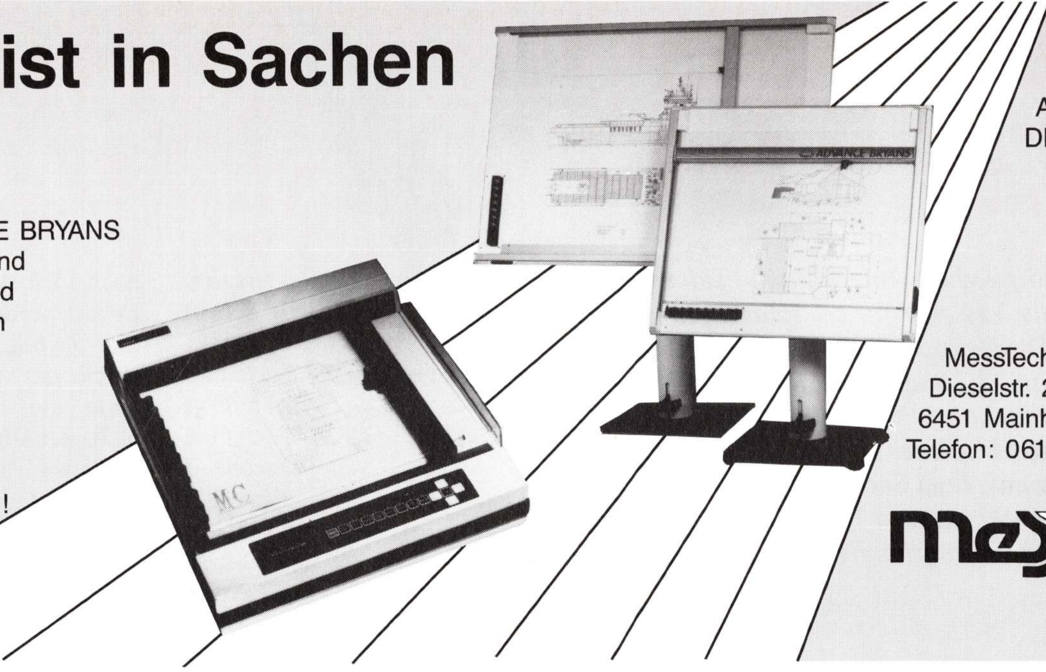


Schlicht und dennoch effektiv: auf HRPD 108 befindet sich ein Videotitler.

Asymmetrie ist machbar

Da SportsText alle Texte mittenzentriert anzeigt, läßt sich ihre horizontale Position nur durch geschicktes Einfügen von Leerzeichen innerhalb der Script-Datei variieren. Da hierbei die Größe der Fonts eine wesentliche Rolle spielt, muß man zunächst etwas experimentieren, um die nötige Anzahl von Leerzeichen zu ermitteln. Muß man diese Methode häufiger verwenden, so empfiehlt es sich, auf kariertem Papier ein ▶

Spezialist in Sachen Plotter



Plotter von ADVANCE BRYANS (Formate A0 – A4) sind HPGL kompatibel und sichern zuverlässigen Betrieb mit allen wichtigen Software-Paketen und Computern.

Fachhändler gesucht!

z.B.
Colorwriter 6120
A4/A3-Rollenplotter
DM 2610,60

MessTech GmbH
Dieselstr. 21
6451 Mainhausen 2
Telefon: 06182-26008

MessTech

I 600

Prg. für alle AMIGA-Modelle
– Exzellent in Struktur, Grafik, Sound – alle Prg. in Deutsch –

ASTROL. KOSMOGRAMM
— Nach Eingabe von Namen, Geburtsort (geografischer Lage) und Geburtszeit werden errechnet: Sternzeit, Aszendent, Medium Coeli, Gestirnestände im Tierkreis, Häuser nach Dr. Koch/Schack (Horoskop-Daten mit Ephemeriden. Außer dem Bildschirmdisplay kann Ausdruck auf 2 DIN-A4-Seiten erfolgen; davon ½ Seite allgemeines Persönlichkeitsbild mit Partnerschaftskriterien und ½ Seite Tierkreisdiagramm (Horoskop). Alle Planeten mit Sonne und Mond. Für alle Berufs- und Hobby-Astrologen eine unentbehrliche Arbeitserleichterung. **78,—**

AMIGA TYPIST
Der AMIGA als Schreibmaschine. Bildschirm-Display — zeilenweiser Druck. Ca. 30 verschiedene Schriften. Kopie-Ausdruck — Textfile auf Disk. **88,—**

BIOKURVEN
Zur Trendbestimmung der Biorhythmen und des seelisch-/geistig-/körperlichen Gleichgewichts mit Druck des Kurvendiagramms von oben nach unten in beliebiger Länge.
In der rechten Blatthälfte das Diagramm, links eine Auswertung des Gesamtpotentials für jeden Tag. Werte für bestimmte Tage auch auf dem Bildschirm. Ausführliche Beschreibung der wissenschaftlichen Grundlagen.
Ideal für Partnervergleiche. **58,–**

GESCHÄFT
– Bestellung, Auftragsbestätigung, Rechnung, Lieferschein, Mahnung, 6 Briefrahmen mit Firmendaten zur ständigen Verfügung (Anschrift, Konten usw., Menge/Preis, Rabatt/Aufschlag, MwSt., Skonto, Verpackung, Versandweg usw.) Mit Einbindung von abgespeicherten Adressen und Artikeln. **198,–**

GELD
– Man wählt mit der Maus unter 25 Rechenroutinen in den Bereichen: Anlage – Kapital – Vermögensbildung – Rentensparen – Rendite – Lasten – Zinsen/Zinseszinsen – Kredit – Hypotheken – Laufzeit – Amortisation – Ratenzahlung – Wertverlust – Nominal- und Effektivzinsen – Ausdruck vollständiger Tilgungsraten – Diskontierung – Devisen/Sorten – Konvertierung **98,–**



I. Dinkler

Am Schneiderhaus 17 · D-5760 Arnsberg 1
Tel. 0 29 32 / 3 29 47

usw. usw. – Fordern Sie mit Freiumschlag unsere Liste an!
Im Computer-Center oder bei uns zu obigen unverbindlich empfohlenen Preisen + DM 3.– bei Vorkasse oder DM 4.70 bei Nachnahme

KALORIEN-POLIZEI – Nach Eingabe von Größe, Gewicht, Geschlecht, Arbeitsleistung erfolgt Bedarfsrechnung und Vergleich m. d. tatsächlichen Ernährung (Fett, Eiweiß, Kohlehydrate), Idealgewicht, Vitalstoffe, auf Wunsch Ausdruck. **58,–**

Inventur, Fibu-gerecht 118,—
Provisionsabrechnung 118,—

Dateiverwaltungen:

Adressen	**69,—**
Bibliothek	**88,—**
Briefmarken	**98,—**
Diskothek	**78,—**
Dokumente	**98,—**
Exponate	**118,—**
Galerie	**118,—**
Lagerartikel	**88,—**
Museum	**118,—**
Personal	**88,—**
Videothek	**78,—**

GIGATRON

NEU! auch als Bausatz

1.8MB AMIGA 500

Die 1.8-MB-Karte ist supereinfach einzubauen:

- als 0.5-MB-Erweiterung auf 1 MB intern: AMIGA umdrehen, Speichererweiterungsklappe öffnen und Karte einstecken – die Garantie bleibt erhalten.
- als 1.8-MB-Erweiterung auf 2.3 MB intern: Gehäuse öffnen, GARY-Chip herausnehmen, Adapter in den GARY-Sockel einsetzen, GARY aufstecken – fertig! **DM 1197,–**

Die 1.8-MB-Karte gibt es jetzt auch

- als Bausatz mit allen Teilen, Schaltplan und Bestückungsliste, jedoch **ohne** 1-MegaBit-Chips (511000) **DM 228,–**
- als Bausatz wie oben, jedoch fertig gelötet mit allen Teilen, ebenfalls **ohne** 1-MegaBit-Chips (511000) **DM 342,–**

NEU!

3.8MB AMIGA 500

Für Unersättliche: Die 3.8-MB-Karte für 4.3 MB Mammutspeicher intern:

- als 1.8-MB-Erweiterung wie oben (bereits fertig mit 1-MegaBit-Chips bestückt) – zusätzlich mit weiteren 16 RAM-Chips (511000) selbst bestücken, einbauen und dann den zusätzlichen Speicher mit „Add-Mem" (40000 – 5FFFF) einbinden – fertig! **DM 1698,–**
- komplett bestückt mit 3.8 MB **DM 2428,–**

NEU!

1.8MB AMIGA 1000

Endlich!
Die 1.8-MB-Karte für den 1000er:

- wird ganz einfach in den vorderen Speichererweiterungsschacht des Rechners gesteckt. Die bisherige 256 KB-Erweiterung wird abgezogen, die neue 1.8 MB-Karte wird aufgesteckt, dann einen Adapter in den CPU-Sockel, CPU aufstecken – fertig. 512 KB bleiben nach dem Abschalten erhalten.
- läuft absolut problemlos mit Sidecar **und** Festplatte **DM 1311,–**

Alle Karten sind bis 1.8 MB autokonfigurierend und mit gesockelten ICs (Ausnahme: Bausatz) sowie einer Echtzeituhr (akkugepuffert) versehen. Sie arbeiten bereits auch unter Workbench 1.3.

Aufgrund der enormen Nachfrage nach 1-MegaBit-Chips auf dem Weltmarkt liefern wir in der Reihenfolge der Bestellungen aus. **Ordern Sie bitte rechtzeitig.** (Die Preise sind unverb. Preisempfehlungen. Techn. Änd. vorbehalten)

Die gigantischen Speicherkarten erhalten Sie nur im guten Fachhandel oder bei

Gigatron G. Preuth, R. Tiedeken
(Entwicklung, Service & Versand)
Resthauser Str. 128, 4590 Cloppenburg
Telefon 04471/3070

FreeCom Wolfgang F.W. Paul
(Auslieferung & Service Raum Hamburg)
Bismarckstraße 2, 2000 Hamburg 20
Telefon 040/49 59 90



Entwurfsraster für jede Schriftart zu entwickeln. Um die Texte nach rechts zu verschieben, genügt es, links des Textes entsprechend viele Leerzeichen einzufügen. Etwas schwieriger ist eine Ausrichtung nach links zu bewerkstelligen, da rechts vom Text stehende Leerzeichen nicht berücksichtigt werden und man gezwungen ist, hinter das letzte Leerzeichen noch ein druckbares Zeichen zu setzen. Damit dieses nicht auf dem Bildschirm erscheint, muß eine überlange Zeile generiert werden, die von SportsText links und rechts abgeschnitten wird. Damit das Abschneiden auf der linken Seite nicht mitten im Text erfolgt, müssen diesem ebenfalls entsprechend viele Leerzeichen vorangestellt werden. Hier ein Beispiel für die Schriftgröße Topaz 8 (Das Fragezeichen in der zweiten Zeile erscheint nicht mehr auf dem Bildschirm):

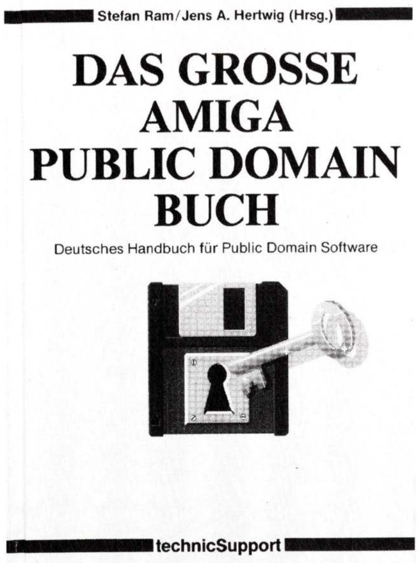
01:(37 Leerzeichen)rechts
02: links(34 Leerzeichen)?

Größere Fonts-Auswahl durch Fremd-Dateien

Sehr schade ist, daß der Autor keine Option zum Einladen weiterer Fonts vorgesehen hat, zumal die verwendeten Standard-Zeichensätze nicht gerade zu den schönsten gehören. Glücklicherweise ist es jedoch möglich, fremde Fonts unter falschem Namen abzuspeichern und dann beispielsweise einen Outline-Zeichensatz unter Diamond 20 abzurufen. Da dieses Umbenennen von Zeichensätzen für einige Verwirrung sorgt, sollte SportsText eine eigene Diskette erhalten, in deren Fonts:-Directory dann nach Lust und Laune manipuliert werden kann. Zu diesem Zweck zieht man eine Kopie der Workbench-Diskette und entfernt einige nicht benötigte Dateien, um für ausreichenden Platz zu sorgen. Anschließend zieht man mit der Maus die Sports Text-Schublade von RPD 108 in das Fenster der neuen Workbench-Diskette. Vom CLI aus kann man nun einen Blick in das Fonts-Verzeichnis der neuen Diskette werfen. Die eigentlichen Fonts finden sich in den Unterverzeichnissen, die den Namen des betreffenden Fonts tragen. Für einen besseren Überblick sollte man dort alle Fontgrößen löschen, die von Sportstext nicht angesprochen werden können. Die nun noch bestehenden Fonts lassen sich nun leicht mit anderen Zeichensatzdateien überschreiben. Um z.B. statt Diamond 20 den Outline 18-Fonts von RPD 63 zu verwenden, kopiert man diesen mit einem Befehl wie copy df1:fonts3/outline/18 to df0:fonts/diamond/20 auf die Sports-Text-Diskette. Nach dem Neubooten des Systems mit dieser Diskette läßt sich durch Anklicken des Diamond 20-Gadgets die Outline-Schrift wählen.

Analog hierzu lassen sich die anderen Fonts umbelegen und die Möglichkeiten von SportsText beträchtlich erweitern. Man kann noch einen Schritt weitergehen und mit dem FontEdit von Fishdisk 030 eigene Fonts erstellen, die neben Buchstaben und Ziffern Symbole und Piktogramme enthalten, wie sie für bestimmte Sportarten typisch sind. Jedoch ist SportsText ein Programm, dessen Einsatz sicher nicht auf das Vertiteln von Sportfilmen beschränkt bleiben muß. *(Ralf Leithaus)*

Hallo, AMIGA-Freunde, laßt doch mal die Mäuse an den Speck!



S. Ram/J. A. Hertwig (Hrsg.)
DAS GROSSE AMIGA PUBLIC DOMAIN BUCH, Band I, ISBN 3-926847-01-8, DM 49,-

AMIGA PUBLIC DOMAIN HANDBUCH, Bd. I und Bd. II

Das GROSSE AMIGA PUBLIC DOMAIN HANDBUCH Bd. I ist der Schlüssel zur Schatztruhe von fast 900 PD-Disketten. Das deutsche Handbuch öffnet die PD-Schatztruhe für viele Anwender. In Band I dieser Reihe sind 44 wichtige PD-Programme mit genauen Anleitungen beschrieben. Z.B. proff, life, AmigaTeX, hack, kermit, units, vt100, amcat, arc, setfont, record, replay, make, PipeHandler, PrintText, MenuBuilder, uShow, DPSlide, PrtDrvGen, blitz, gfxmem, disksalv und viele andere Programme.

Darüber hinaus bringt Band I: Einführung in die PD-Software, CLI-Hilfen, Erste Schritte mit PD, Beschreibung zum Editor ED. Und: die komplette Liste der wichtigsten PD-Reihen

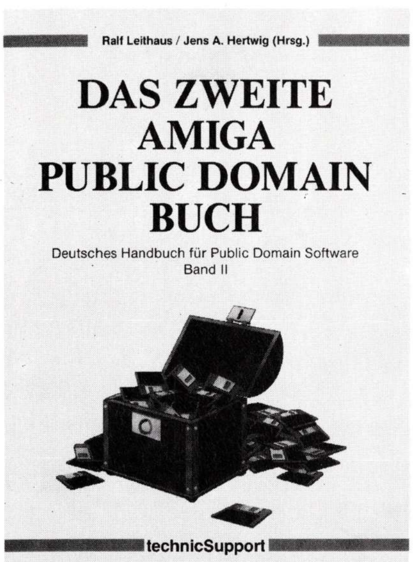
Die Chance für AMIGA-Einsteiger:

Public Domain Software zu Superpreisen!!

technicSupport Marketing und Verlag GmbH, Bundesallee 36-37, 1000 Berlin 31

NEU! Sofort lieferbar: Bd. II

DIE FORTSETZUNG!



Hrsg. R. Leithaus/J. A. Hertwig
DAS ZWEITE AMIGA PUBLIC DOMAIN BUCH
ISBN 3-926847-05-0, DM 49,-

Band II der PD-Reihe von technicSupport bringt z.B. deutsche Anleitungen zu folgenden Programmen:
Bankn (Verwaltung Girokto.), **Freud** (Psychotest), **WordWright** (Supertextverarbeitung), **C-Light** (Ray-Tracing), **mCAD** (Konstruktionsprogramm), **GOMF** (Ende des Guru!), **MRBackup** (für Festplatten), **Turbo Backup** (bestes Kopierprogramm), **VirusX** (automatischer Virusdetektor), **M2Amiga** (Modula 2) und Spiele, Grafik, Sound, Animation. Infos zur Workbench 1.3 und 1.4 sowie zur RPD-Reihe. 46 Programme beschrieben!

NEU! NEU! NEU! Ab November!
DAS GROSSE AMIGA SPIELE BUCH
ISBN 3-926847-02-6, DM 49,-
Subscriptionspreis bis 1.11.1988: DM 39,-

BESTELLUNG Ich bestelle bei technicSupport GmbH, Bundesallee 36-37, 1000 Berlin 31, T. 030-8621314(15)(99)

....x GROSSE AMIGA PUBLIC DOMAIN BUCH, Band I, ISBN 3-926847-01-8	DM 49,-
....x AMIGA PUBLIC DOMAIN BUCH, Bd. II ISBN 3-926847-05-0	DM 49,-
....x Band I und Band II zum Kombi-Super-Preis	DM 89,-
....x 10 Disketten mit 44 Programmen zu Band I	DM 95,-
....x alle Disketten mit 46 Programmen zu Band II	DM 95,-
....x Kombi-Angebot Band I und 44 Programme	DM 136,-
....x Kombi-Angebot Band II u. 46 Programme	DM 136,-
....x Spar-Angebot: Bd. I und Bd. II und 21 Disks	DM 249,-
....x SPIELE BUCH, ISBN 3-926847-02-6 (39,-)	DM 49,-

Je + DM 5,- Versandkostenanteil. Ausland nur DM-Scheck.
Ich bezahle per .... Nachnahme, per .... Verrechnungsscheck

Name: ....

Straße: ....

Ort: ....

Datum: .... **Unterschrift**



Fischers Fritz fischt frische...

Mit großer Regelmäßigkeit sorgt Fred Fish für den Ausbau seiner Software-Bibliothek. Brandneu sind die Fishdisks 147 bis 154. Der Fang hat sich wieder einmal gelohnt. Die Ausbeute reicht vom Editor bis zu verspielten Icons.

Fishdisk 147 liefert einen guten Grund, den Standard-Editor Ed endlich einmal über Bord zu werfen: Sie enthält die neueste Version (MG 2b) des „MicroGNUEmacs", der allein schon durch seine Geschwindigkeit beim Scrollen, Suchen und Einfügen überzeugt. Während der Editor selbst in einer lauffähigen Version vorliegt, wurden alle Anleitungen und Source-Codes mit dem Programm „Zoo" (Fishdisk 136) komprimiert. Mit diesem Programm müssen die Dateiarchive zunächst unter Verwendung der „-extract"-Option wieder dekomprimiert werden. Besitzer von AmigaTeX dürfen sich übrigens freuen: Sie können aus der Datei „mg.tex" (im LaTeX-Format) ein professionell gestaltetes Manual gewinnen.

Disk 148 enthält mit „EFJ" (Escape from Jovi) ein kleines Geschicklichkeitsspiel, bei dem man ein Raumschiff durch ein Höhlenlabyrinth steuern muß. „HandyIcons" ermöglicht das Aufrufen von Programmen über ein neues Menü auf der Workbench, „Scrambler" das Chiffrieren und Dechiffrieren von Texten. „Fme" ist ein Editor für das Spiel FirePower, der jedoch auf einer separaten Diskette installiert werden muß, um nicht den Guru zu wecken. Fred hat den Installationsvorgang in der Datei „Readme.fnf" ausführlich beschrieben.

Für DFÜ-Freunde dürfte das Programm „Keep" von Fishdisk 149 von Interesse sein. Es dient zum Sichten und Bearbeiten längerer Dateien, wie man sie aus den verschiedenen Netzwerken „downloaden" kann. Bei „DX-VoiceSorter" handelt es sich um ein Hilfsprogramm zur Bearbeitung der VoiceFiles des Voicefiler-Programms (Disk 82) für den Yamaha DX-7-Synthesizer. „Scheme" ist ein Dialekt der LISP- Programmiersprache, „Less" ein Update zu Fish 92: Es wurden nicht nur einige Bugs beseitigt, Less arbeitet nun auch im Interlace-Modus und verfügt über eine Print-Funktion. In der „Animal-Sounds"-Schublade findet man nicht nur digitalisierte Tierstimmen, sondern auch ein passendes Abspielprogramm namens „ssp".

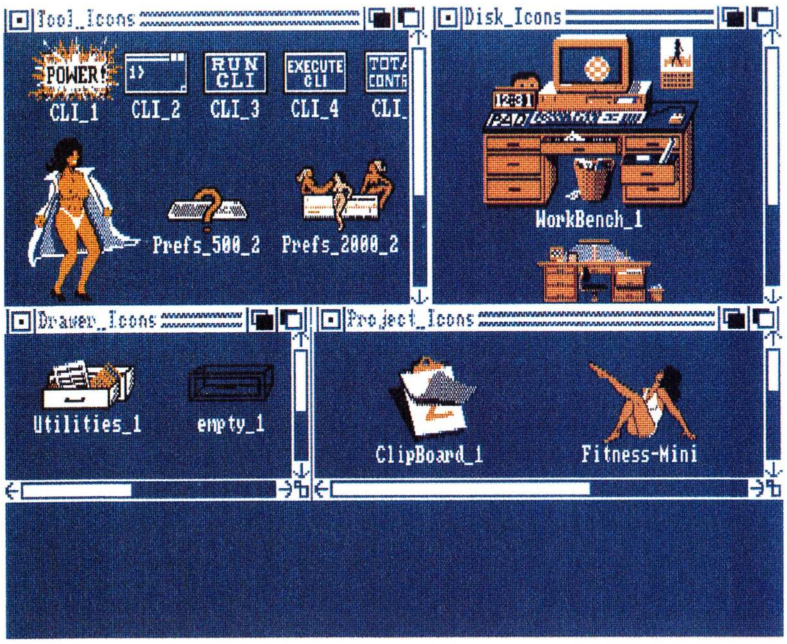
Fishdisk 150 offeriert eine an Apple- und PC-Pilot angelehnte Version dieser Programmiersprache, verbunden mit „DEMONPS.pil", einer Demo für interaktive Informationsvermittlung an Bildschirmgeräten. „Airfoil" zeigt Luftströmungsverläufe über Tragflächenprofilen. Wer sich mit dieser Materie auseinandersetzen möchte, findet in den Begleittexten ausführliche Informationen über die mathematischen Grundlagen und ihre programmtechnische Umsetzung. „DC10" ist ein Basic-Flugsimulator, „ExecLib" liefert ein Beispiel zur Erstellung eigener Disk-residenter Libraries für Lattice C. Mit „Iconizer" läßt sich eine ganze Bibliothek verschiedener Mauszeiger anlegen, die durch einfaches Anklicken von Icons aktiviert werden können. Abgerundet wird diese Diskette durch das Programm „SteelMemBoot". Es installiert im Bootblock eine Routine, die beim Booten automatisch das Fast-Memory ausblendet.

Neben einer „SlideShow" nur für LoRes-Bilder bietet Fishdisk 151 eine umfangreiche, nach Typen geordnete Sammlung origineller „Icons". Die Grafikdemo „Globe" zeigt eine sich drehende Erdkugel, „SCT" erlaubt das Setzen der Bildschirmfarben vom CLI aus, „Surveyor" zeigt die exakte Position des Mauspointers und kann zum Beispiel Programmierern bei der Plazierung von Gadgets oder anderen Symbolen behilflich sein. „Pcopy" ist ein nicht gerade schnelles Kopierprogramm für exakt zwei Laufwerke. Das Besondere: Auf Wunsch startet Pcopy automatisch, sobald sich beide Disketten im Laufwerk befinden.

C-Programmierer finden auf Disk 152 mit „Blk" ein Tool zum Erstellen von Requestern, während Usenet-Fans sich über „UUCP", die Amiga-Version des Unix to Unix Copy Program, freuen dürfen. Schließlich gibt es noch ein Update zu „Runback" von Fishdisk 65. Die neue Version berücksichtigt endlich auch den aktuellen Suchpfad.

Wer einen HP-Laserjet+ oder einen kompatiblen Drucker besitzt, kann diverse Parameter über die Menüleiste einstellen, wenn er „HPMan" von Fishdisk 153 verwendet. Darüber hinaus enthält diese Diskette die Emulation eines programmierbaren Taschenrechners, die Version 1.3 des „Dme"-Editors und „Synthemania", ein Musikstück, das jedoch nur mit 512 KB oder nach dem Ausblenden des Fast-Rams läuft.

Sehr vielfältig ist das Angebot von Disk 154. Neben sehenswerten Laufschriften im „AssemblyDemos"-Verzeichnis finden sich in der Utilities-Schublade nützliche Hilfsprogramme: „Undelete" rettet versehentlich gelöschte Dateien,



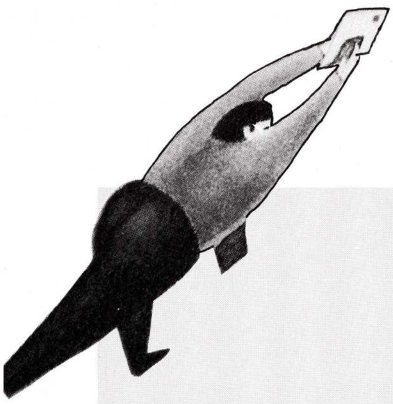
Selbst ernsthafte Programme bekommen mit witzigen Icons Pfiff.



Auf Fish 150 befinden sich Mauszeiger, die nach Anklicken sofort aktiv sind.

„WhereIs“ sucht nach Dateien bzw. Directories, „CAL“ zeigt ein Kalenderblatt für den aktuellen oder gewünschten Monat, „DClock“ ist eine kleine Zeit- und Speicherplatzanzeige, die bequem in die Menüleiste paßt. Im „DiskLib“-Verzeichnis finden sich zwei Hilfen für die Katalogisierung von Disketten: „Lib“ gibt die Inhalte einer Diskette aus, während „ToLib“ Freds „readme-.dist“- bzw. „Contents“-Dateien in eine von „Lib“ verwertbare Form konvertiert. Die Programme „Print“ und „Spool“ erlauben das Ausdrucken von Texten als Hintergrundprozeß, so daß der Rechner für weitere Aufgaben frei bleibt. „Wicon“ erlaubt es, Fenster in Icons zu verwandeln und sie durch Anklicken eben dieser Icons wieder zu öffnen. Den inhaltlichen Schwerpunkt von Fishdisk 154 bildet jedoch die Virenbekämpfung. Für diese Aufgabe stehen gleich drei Programme zur Verfügung, die unterschiedliche Wege beschreiten. „VirusX“ untersucht den Bootblock jeder neu eingelegten Diskette und den Speicher. Die neue Version 1.21 erkennt sowohl das SCA- als auch das Byte Bandit-Virus.

„Guardian“ installiert sich beim Aufruf resetfest im Speicher und löst anschließend (Vorsicht!) einen Reset aus. Nun zeigt Amiga ein verändertes Einschaltbild: Unter der Hand mit der Workbench-Diskette informiert der Schriftzug „Guardian v 1.1“ über die Aktivität des Detektors. Nach jedem Bootvorgang informiert „Guardian“ jetzt in einem riesigen Alert-Fenster über mögliche Gefahren. Sehr positiv, daß auch der Bootblock angezeigt wird, wie man es von einem Diskettenmonitor gewohnt ist. Besitzer eines Amiga 1000 können „Guardian“ mit dem Hilfsprogramm „Creator“ in ihre Kickstart-Diskette integrieren. Weniger geglückt scheint mir „Virus-Alert!“. Dieses Programm installiert einen neuen Bootblock (schon das ist nicht ungefährlich!), der sofort nach dem Booten einen Alert hervorruft. Bleibt dieser Alert einmal aus, dann ist der Bootblock in irgendeiner Form (von einem Virus?) überschrieben worden. Eine Alarmmeldung, die gerade dann ausbleibt, wenn Gefahr droht, ist wohl nicht mit den Prinzipien menschlicher Informationsverarbeitung in Einklang zu bringen! *(Ralf Leithaus)*



Dme — Das Chamäleon unter den Editoren

Ein guter Editor gehört in den „Werkzeugkasten“ eines jeden Computerbesitzers. Was aber zeichnet einen guten Editor aus? Einigkeit besteht zumindest in der Forderung nach hoher Arbeitsgeschwindigkeit, d.h. dem schnellen Umsetzen der Eingaben (Scrollen, Texteinfügen, Blockverschieben usw.) in das entprechende Resultat auf dem Bildschirm. Selbstverständlich ist auch, daß er über alle grundlegenden Funktionen zur Textbearbeitung verfügen muß, möglichst auch über die deutschen Sonderzeichen. Anders verhält es sich bei der Frage nach der idealen Benutzerführung: Hier spielen individuelle Vorlieben eine zentrale Rolle. Gerade in diesem Punkt zeigt sich Dme von Fishdisk 153 anderen Editoren überlegen, erlaubt er doch eine weitestgehende Anpassung der Arbeitsumgebung an die Wünsche seines Benutzers. Aber auch die Grundanforderungen bewältigt Dme mühelos, er ist sogar schneller als die bekannten Blitzfonts.

Nach dem Anklicken des Dme-Icons öffnet sich zunächst ein leeres Fenster, mit dem sich ohne Studium des Manuals recht wenig anfangen läßt. Ein Klick in die Menüleiste stiftet nur Verwirrung, da das Dme-Fenster dadurch lediglich in ein Icon verwandelt wird. Erst nach der Einrichtung eines Menüs durch den User erhält die rechte Maustaste ihre ursprüngliche Funktion zurück, solange dient die Menüleiste lediglich als Statusanzeige. Die Hauptrolle spielt die Esc-Taste, mit der man in eine Kommandozeile gelangt, in der sämtliche Funktionen in Form von Befehlswörtern eingegeben und mit der Return-Taste abgeschlossen werden. Fast alle Funktionen lassen sich jedoch auch über Tastaturäquivalente aufrufen, die jeweils in Klammern angegeben werden.

Diskettenoperationen erleichtert

Die einfachste Methode, eine Datei zu laden, ist das Kommando NEWFILE (F2), gefolgt vom Dateinamen einschließlich der kompletten Pfadangabe. SAVEOLD (F9) speichert die Datei unter dem aktuellen Namen, der auch in der Titelleiste angezeigt wird, SAVEAS ermöglicht das Speichern unter einem beliebigen Namen. Wesentlich komfortabler geht es, wenn sich die „Arp.library“ (Panorama 65) im LIBS:-Verzeichnis befindet: Jetzt lassen sich mit ARPLOAD und ARPSAVE die gewohnten Dateiauswahl-Requester aufrufen. Für das Einfügen einer Datei in den bestehenden Text steht der Befehl INSFILE (F1) zur Verfügung.

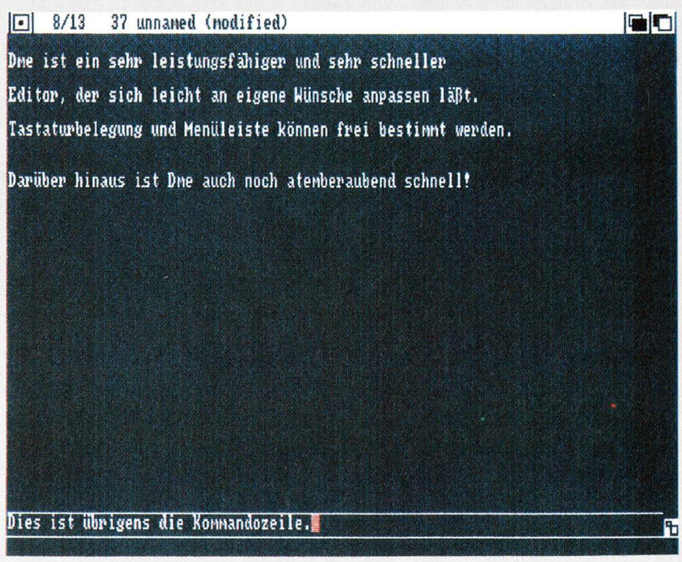
Bewegen im Text

Neben den üblichen Steuertasten, wie man sie von jeder Textverarbeitung kennt, hält Dme viele weitere Möglichkeiten bereit. GOTO n (Ctrl-G) springt in die genannte Zeile, GOTO +/- n um die angegebene Zeilenanzahl vor- bzw. rückwärts, TOP (Shift-Up) springt an den Anfang, BOTTOM (Shift-Down) an das Ende des Textes. Mit PAGEUP (Ctrl-Up) und PAGEDOWN (Ctrl-Down) blättert man eine Seite auf- bzw. abwärts. SCROLLUP (Alt-Up) und SCROLLDOWN (Alt-Down) scrollen den Screen, ohne daß die Cursorposition verändert wird. Mit WLEFT (Ctrl-L) und WRIGHT (Ctrl-R) wird der Cursor auf den Beginn des voranstehenden oder folgenden Wortes gesetzt. Bei größeren Entfernungen empfiehlt es sich, den Cursor mit der linken Maustaste zu plazieren. Für das Löschen einer kompletten Zeile gibt es das Kommando DELINE (Shift-Del), für das Löschen vom Cursor bis zum Zeilenende den Befehl REMEOL (Ctrl-Del).

Anfang und Ende eines Textblocks werden markiert, indem der Cursor auf die entsprechenden Stellen plaziert und anschließend der Befehl BLOCK (Ctrl-B) aufgerufen wird. Ein so markierter Block erscheint in schwarzer statt in weißer Schrift. Er kann mit BCOPY (Alt-C) vor die aktuelle Zeile verschoben, mit BDELETE (Alt-D) gelöscht oder mit BSAVE (F7) gespeichert werden. Mit UNBLOCK (Ctrl-U) läßt sich eine Blockmarkierung wieder aufheben.

Suchen und Ersetzen

Über die Befehle FINDSTR und REPSTR lassen sich



Editorfläche und Kommandozeilen präsentieren sich bei Dme separat.

Such- und Ersatzbegriff festlegen, ohne einen Suchvorgang auszulösen. Mit NEXT (Ctrl-N) und NEXTR (Alt-R) läßt sich der Text in Vorwärtsrichtung durchsuchen, NEXTR bewirkt dabei ein automatisches Ersetzen. Dasselbe gilt für PREV (Ctrl-P) und PREVR im Rückwärtsgang. Das Suchen läßt sich auch durch FIND <Suchbegriff> (Ctrl-/), das Ersetzen durch FINDR <Suchbegriff> <Ersatzbegriff> auslösen.

Weitere Features

Der voreingestellte Einfügemodus läßt sich mit INSERTMODE OFF (Ctrl-O) aus-, mit INSERTMODE ON (Ctrl-I) wieder einschalten. Die WordWrap-Funktion läßt sich über WORDWRAP TOGGLE (Ctrl-W) steuern. Mit EXECUTE können sogar CLI-Befehle aufgerufen werden. Insgesamt verfügt Dme über zirka 100 Befehle, die nicht alle in diesem Artikel erläutert werden können. Zu ihnen gehören das Setzen von Tabulatoren, das Öffnen und Reskalieren weiterer Fenster, Bedingungsabfragen, Einbinden von ARexx- Makros und anderes mehr. Näheres erfährt man aus der Datei „dme.doc“ im „dme/docs“-Verzeichnis der Fishdisk.

Die herausragende Fähigkeit des Dme besteht darin, sich wie ein Chamäleon verwandeln zu können. Der Benutzer kann alle Tastaturäquivalente nach seinen Vorstellungen umbelegen und sogar eine ganz persönliche Menüleiste einrichten. Die hierzu erforderlichen Anweisungen müssen in einer Datei namens „.EDRC“ abgelegt werden, die im S:-Verzeichnis oder in demselben Verzeichnis wie Dme selbst stehen muß. Innerhalb dieser Datei muß jede Anweisung in einer eigenen Zeile stehen. Für die Umbelegung der Tastatur ist der „MAP“-Befehl zuständig. Ihm müssen die Abkürzung der Tastenkombination und der gewünschte Befehl nachgestellt werden. Hier die wichtigsten Abkürzungen: a (Alt), A (Amiga), s (Shift), c (Ctrl). Für die Übergabe von Argumenten ist der ESCIMM-Befehl zu verwenden, der einen Wechsel in die Kommandozeile bewirkt und diese mit einem beliebigen Text vorbesetzt. Zur Verdeutlichung ein kurzes Beispiel:

```
map c-l deline
map s-a top
map s-e bottom
map a-l (escimm (newfile ))
```

Nach dieser Umbelegung bewirkt Ctrl-L das Löschen der aktuellen Zeile, Shift-A einen Sprung an den Anfang, Shift-E dagegen an das Ende des Textes. Möchte man eine neue Datei laden, drückt man Alt-L und braucht in der Kommandozeile nur noch den Dateinamen einzugeben.

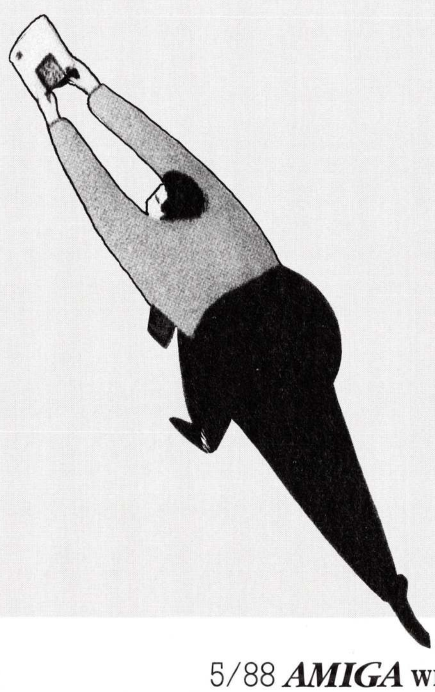
Ganz analog läßt sich die Menüleiste einrichten. Das folgende Beispiel bedarf keiner ausführlichen Erklärung, die verwendeten Befehle sprechen für sich. Die Menüpunkte für „Laden“ und „Speichern als“ setzen natürlich wieder die „Arp. library“ voraus.

```
menuadd Datei Laden arpload
menuadd Datei Speichern saveold
menuadd Datei (Speichern als) arpsave
menuadd Datei Beenden quit
menuadd Cursor Textanfang top
menuadd Cursor Textende bottom
menuadd Cursor (Gehe zu Zeile...)
(escimm(goto ))
menuon
```

Zum Austesten solcher Anweisungen lassen sich diese Befehle übrigens auch über die Kommandozeile eingeben.

Abschließend noch einige Tips zur Gestaltung der Arbeitsumgebung. Alle häufig benutzten Befehle sollten in jedem Fall über die Tastatur erreichbar sein. Dabei ist es Geschmackssache, ob sich die Abkürzungen an deutschen oder englischen Begriffen orientieren, ob also Ctrl-L für Löschen steht oder Ctrl-D. Beide Möglichkeiten sollten jedoch nicht miteinander vermischt werden. Empfehlenswert ist es, die Selektiertasten bestimmten Funktionsgruppen zuzuordnen, zum Beispiel die Shift-Taste für Diskettenoperationen, die Alt-Taste dagegen für Bewegungen im Text zu benutzen. Da der Griff zur Maus immer etwas Zeit kostet, sollten in der Menüleiste nur seltener benutzte Funktionen Platz finden. Sie kann aber auch zur Anzeige bestimmter Funktionen und ihrer Tastaturäquivalente herangezogen werden und so die fehlende Help-Funktion ersetzen.

(Ralf Leithaus)



Trickkiste für Programmierer

Es gibt Probleme, die sich mit einem winzigen Trick lösen lassen. Wer einen weiß und nennt, und so dem Leser weiterhilft, wird mit mindestens 50 Mark belohnt.

$025 Spritebewegungen mit reduziertem Flackern

Will man Sprites (oder Bobs) animieren, muß man mit OBJECT.SHAPE zwei Objekte immer wieder austauschen. Da dabei die Objekte aber immer wieder auf die linke obere Bildschirmecke zurückgesetzt werden, entstehen zwischen den beiden Bewegungsphasen kleine Pausen, wodurch die Objekte noch mehr flackern, als sie es sonst schon tun. Etwas besser sieht das Ganze aus, wenn man das Objekt für die erste Bewegungsphase so lange auf dem Bildschirm stehen läßt, bis das Objekt für die zweite Bewegung zu sehen ist und das erste erst dann ausschaltet.

```
Schrittweite=10
ErstesSprite=1
c=0
OPEN "Sprite1" FOR INPUT AS #1
  OBJECT.SHAPE ErstesSprite,INPUT$(LOF(1),1)
CLOSE 1

OPEN "Sprite2" FOR INPUT AS #1
  OBJECT.SHAPE ErstesSprite+1,INPUT$(LOF(1),1)
CLOSE 1

OBJECT.X 1,600
OBJECT.X 2,600
OBJECT.Y 1,100
OBJECT.Y 2,100

FOR x=600 TO 0 STEP -Schritweite
  OBJECT.ON c+ErstesSprite
  OBJECT.OFF 1-c+ErstesSprite
  FOR n=Schrittweite TO 1 STEP -1
    OBJECT.X ErstesSprite,x+n
    OBJECT.X ErstesSprite+1,x+n
  NEXT
  c=1-c
NEXT
```

Die Variable „ErstesSprite“ gibt die Nummer des Objektes an, das für die erste Bewegungsphase steht. Mit c wird festgelegt, ob die Animation mit der ersten (c = 0) oder der zweiten (c = 1) Bewegungsphase beginnt. Die Schrittweite gibt an, wie weit sich das Sprite bewegt, bevor die nächste Bewegungsphase beginnt.

(Steffen Frank)

$026 Fehler im ObjEdit

Der zum AmigaBasic beigelegte Object-Editor speichert Sprites in den falschen Farben ab. Damit das Weiß nicht mehr nach Hellgrün aussieht, muß man in dem Unterprogramm „SaveFile“ in der Zeile PRINT #1,MKI$(&HFF) das &HFF in &HFFF umändern.

(Steffen Frank)

$027 Mehrere Files gleichzeitig zum Drucken geschickt

Wenn man mit 'Run Type File_1 To PRT: einen Druckauftrag zum Drucker geschickt hat, und noch einen Druckauftrag mit Type File_2 To PRT: abschicken will, wird dieser vom Amiga mit der Fehlmeldung „can't open prt:“ abgelehnt. Wer aber dennoch mehrere Files zum Drucken abschicken will, muß sich eines Tricks bedienen. Der Join-Befehl ermöglicht das Zusammenfügen von mehreren Files zu einem. Wird aber als Zielfile der Drucker (PRT:) angegeben, so werden die Files nacheinander ausgedruckt.

```
Join File_1 File_2 File_3 ... File_n As PRT:
```

(Thomas Meidinger)

$028 Tips zum Schreiben von kurzen Files

Will man schnell eine kurze Batch-Datei (oder ein Script-File für den Changer) schreiben, dann lohnt es sich nicht den „Ed“ einzuladen, da das Problem auch durch einen Befehl gelöst werden kann. Dabei wird die Standard-Ausgabe in ein File umgelenkt. Zu beachten ist, daß zum Beenden des Files EOF (Ctrl + \) gedrückt werden muß.

Copy * To filename

(Thomas Meidinger)

$029 Zahlen ohne Spacezeichen

Wird in Basic mit PRINT a eine Zahl ausgegeben, steht vor und hinter der Zahl jeweils ein Leerzeichen. Das wirkt sich besonders störend aus, wenn eine Zahl direkt hinter einem Text stehen oder zum Beispiel in einem Kasten ausgegeben werden soll, dessen Kanten nicht von den Leerzeichen überschrieben werden sollen. Diese Leerzeichen können entfernt werden, wenn man die Zahl in einen String umwandelt und dann die Spaces entfernt:

```
        a$=STR$(a)
        a$=RIGHT$(a$,LEN(a$)-1)
        PRINT a$
oder vereinfacht:
        a$=RIGHT$(STR$(a),LEN(STR$(a))-1)
        PRINT a$
```

(Steffen Frank)

C als erste Fremdsprache

Einsteigerkurs, 1. Teil

Mit einer schnellen Programmiersprache wird der 68000er zu einer wahren Wundermaschine. Für längere Aufgaben ist die Hochsprache C das ideale Programmierwerkzeug — schneller und effektiver als BASIC, strukturierter und komfortabler als Assembler.

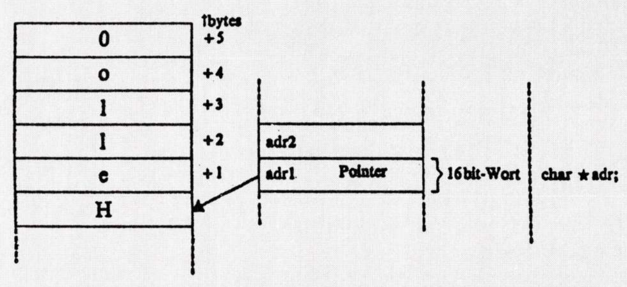
Die Muttersprache des AMIGA ist C, ein großer Teil des Betriebssystems wurde damit geschrieben. Mit dem entsprechenden Know-how lassen sich auch Eigenentwicklungen mit C effektiv realisieren. Schwierigkeiten bereitet jedoch der Einstieg in C, besonders, wenn man die Spaghettistruktur des C64-BASIC gewöhnt ist. C wartet nicht nur mit einer Reihe neuer Begriffe auf, sondern verlangt vom User vor allem strikte Einhaltung der sehr rigiden Regeln. Diese Programmierdisziplin ist wie bei Pascal und Fortran eine besonders hohe Hürde für Anfänger.

Die Sprache C wurde vor fast 20 Jahren von Dennis M. Ritchie entwickelt und lief auf den ersten Mainframes (Großrechner wie PDP /7). Sie wurde als Kernsprache des Mehrbenutzersystems UNIX eingesetzt und fand so weite Verbreitung. Heute ist C eine der populärsten Sprachen auf Mikrocomputern. Ritchie suchte eine Programmiersprache, die die Maschinennähe von Assembler und die strukturierte Programmierung von Pascal vereinigt. Das Resultat dieser Überlegungen, sie kennt nur 30 (!) Befehle und ist doch die flexibelste Sprache für Microcomputer. Diese große Freiheit, die C dem Programmierer läßt, ist jedoch gleichzeitig der Stolperdraht für Anfänger. Nur wer sich streng an die Regeln hält, kann darauf hoffen, einmal effektive C-Programme schreiben zu können.

Wiedersehen mit alten Bekannten

Aus der guten alten BASIC-Zeit weiß man, daß ein Programm mit Variablen arbeitet. Doch im Gegensatz zu BASIC muß man diese selbst anfordern und deklarieren. Es gibt die Standardwerte:

char	8 Bit
int	16 Bit auf dem AMIGA! float 32 Bit
double	64 Bit



Pointerzugriff auf Speicherplätze

Zusätzlich können noch *short*, *long* und *unsigned* vorgestellt werden, die dann den Speicherbedarf einer Variablen verändern. Diese Variablen können verschiedene Gültigkeitsbereiche besitzen:

lokale Variablen	innerhalb eines Programmteiles
globale Variablen	innerhalb des gesamten Programms
externe Variablen	auch außerhalb eines Programms

```
                                  OpenLibrary() geholt */
if(( wind=(struct Window *)OpenWindow(&nw))==NULL) exit();
/* · oeffnet Fenster
```

```
 *    falls kein Fehler vorliegt
    sonst Ausstieg*/
/* geschlossen wird das Fenster mit * CloseWindow(wind);
   ....*/
|      LL
```

auto
extern
static
register
Diese Anweisungen werden klein geschrieben, um sie von den vordefinierten Deklarationen wie BOOL, TRUE, FALSE, UWORD zu unterscheiden. Daneben besteht die Möglichkeit, eine Variable in ein Datenregister der CPU (der 68000 hat 8 Daten- und 8 Adreßregister) zu legen. Dazu wird der Ausdruck *register* vorgestellt. Der selten benutzte Ausdruck >static< bewirkt, daß eine Variable nicht nach ihrem Aufruf gelöscht wird. Ob eine Variable vorzeichenbehaftet sein soll oder nicht muß vorab definiert werden. Dies geschieht mit den Ausdrücken: *signed* und *unsigned*
Es ist wichtig, den Wertebereich einer Variablen vorher genau zu beschreiben, um nicht durch Wertebereichsverletzungen einen Absturz zu riskieren. Zusätzlich bietet C die Möglichkeit einen eigenen Variablentyp zu definieren: typedef char ZEICHEN; typedef int INTEGER; Diese definierten Ausdrücke müssen groß geschrieben werden um sie im Programmtext als vordefiniert erkennen zu können. Diese Anweisungen gehören zum Wortschatz der Sprache C, sind aber mit den eigentlichen Befehlen gleichzusetzen. Kontrollstrukturen:
Als eigentliche Befehle gibt es in C:
if *else-if* *switch* *for*
while *do-while* *break* *continue*
goto
Das sind alle Kontrollstrukturen in C und sie reichen tatsächlich aus.
Wie im folgenden Beispiel wird eine „if"-Anweisung in C als Funktionsaufruf der Funktion *if()* programmiert. Sämtliche Befehle werden als Funktionen programmiert; dies verbessert die Lesbarkeit und Gestalt der C-Programme. Die „if"-Anweisung verzweigt, wenn ihr Funktionsinhalt wahr wird, das heißt, wenn die Verzweigungsbedingung erfüllt ist.

```
if(a==b) printf("gleich!\n");
else printf("nicht gleich!\n);
```

zeigt den Aufbau der Verzweigung: Wenn der Wert der Variablen a gleich dem der Variablen b ist verzweigt das Programm in die Anweisung hinter der "if„ Anweisung. Durch geschweifte Klammern können hier auch mehrere Anweisungen zusammengefasst werden:

```
if(a==b) { test=test+1;
           var= a-b;
           printf("mehrere Anweisungen \n);
         }
```

Wenn eine „if"-Anweisung nicht erfüllt wurde, so kann in der „else"-Anweisung erneut verzweigt werden:

```
if(a=b) vergl();
                                OpenLibrary() geholt */
if(( wind=(struct Window *)OpenWindow(&nw))==NULL) exit();
/*   oeffnet Fenster
 *   falls kein Fehler vorliegt
    sonst Ausstieg*/
/* geschlossen wird das Fenster mit * CloseWindow(wind);
   ....*/
|
else if(a > b) printf("groesser ! \n);
```

In der ersten "if„-Anweisung stand die Funktion *vergl()*. Es kann in der "if„-Anweisung auch in ein Unterprogramm gesprungen werden.
Die "for„-Anweisung erlaubt Schleifen mit einer Laufvariablen, die von der Funktion *for()* errechnet wird. Die *for()*-Funktion benötigt drei Parameter, die Anfangsbedingung, Abbruchbedingung und Kriterium festlegen. Beispiel:
for(i=0 ; i=999 ; i+ +) doIT();
bedingt 1000 Schleifendurchläufe(0 ... 999), denn die Anfangsbedingung ist *i = 0* und die Abbruchbedingung ist *i = 999*, wobei pro Schleifendurchlauf i um eins erhöht wird. Wenn keine Parameter angegeben werden liefert *for(;;)* eine Endlosschleife.
Die "while„-Anweisung erlaubt Schleifen, die über ein Kriterium gesteuert werden. Dieses Kriterium prüft, ob die Schleife erneut durchlaufen oder ob das Programm fortgesetzt wird. Beispiel:

```
while(event == NULL)           /* kein Ereignis         */
            {
            warte();
            i=i++;
            if(i>1000) break;  /* Abbruch wenn zu lange */
            } L
```

Die "switch„-Anweisung ermöglicht die Auswahl aus einer Vielzahl von Möglichkeiten. Dabei wird der Programmblock ausgeführt, der durch die "switch„-Anweisung angesprungen wird:

```
switch(schalter) {
            case '1': printf("Fall Nr. 1 \n ");
            case '2': return();
            case '3': abgleichen();
            ...
            }
```

Rechnenvorgänge und Vergleiche und auch komplizierte Bitoperationen werden mit Operatoren möglich gemacht:

+, -, *, /	plus, minus, mal, durch
%	modulo - Division
>, >=, =<, <,	größer, größergleich, kleinergleich, kleiner
==, !=,	ist exakt, ist nicht
=	Zuweisung eines Wertes (!)
++, --	Inkrement und Dekrement
&, !, ^ , <<, >>	Bit - Operatoren

Somit kann man jetzt mit Variablen rechnen:

```
a = b + 2 ; s = fahr++ ; mod = teil % div ;
```

sind gültige Ausdrücke.

```
                              OpenLibrary() geholt */
if(( wind=(struct Window *)OpenWindow(&nw))==NULL) exit();
/*   oeffnet Fenster
 *   falls kein Fehler vorliegt
    sonst Ausstieg*/
/* geschlossen wird das Fenster mit * CloseWindow(wind);
   ....*/
```

Mehr Spielraum auf Feldern

Natürlich lassen sich in C auch Felder definieren und sogar, und das ist eine Besonderheit, während des Programmablaufs erstellen. Ein Feld läßt sich mit einer Warteschlange an einer Kaufhauskasse vergleichen. So wie dort die Kunden hintereinanderstehen, sind auch die einzelnen Inhalte in ▶

einem Feld geordnet. Dies gilt auch bei mehrdimensionalen Feldern wie das folgende Beispiel zeigt:

```
int  i;
int  feld a,b,c  ;
char text[a][b][c]  ;


text[0][0][0] = anfang;
for(i=0; i=20; i++) [
            text[i][0][0] = i;    /* Füllen mit Werten */
```

Solche Arrays finden vielfach Verwendung in der Datenverarbeitung, wenn bestimmte Zellen abgefragt werden sollen, zum Beispiel in Spreadsheats. Die Variablen a,b,c sind die Indexvariablen ihres Feldes und ermöglichen den Zugriff auf bestimmte Elemente des Feldes (Arrays).

Variablen sind teamfähig

Neben den normalen Variablen gibt es die Möglichkeit mehrere Variablen verschiedener Typen zusammenzufassen. Diese Möglichkeit wird jeden Programmierer auf dem AMIGA verfolgen; es sind die Strukturen und Unions:

```
struct SimpleSprite [
            UWORD *posctldata;
            UWORD height;
            UWORD x,y;
            UWORD num;
            ];
```

In diesem AMIGA-typischen Beispiel sind fünf verschiedene Variablen zusammengefaßt, die auch unterschiedliche Typen haben dürfen (int, char,...). *Unions* sind auf dem AMIGA weniger gebräuchlich, sind aber ähnlich aufgebaut wie Strukturen. Diese *Structures* bilden Datenblöcke mit fester Struktur deren Elemente immer den selben Abstand von der Anfangsadresse im Speicher haben. Dies macht sich der Compiler zunutze, wenn er die Adressen von solchen Elementen errechnet. Die *Structures* werden gefunden, indem sie mit einem Namen deklariert werden:

```
struct SimpleSprite simplename;
```

Anhand dieses Deklarators können die Elemente angesprochen werden. Die zu einer Structure gehörenden Elemente werden mit dem Zuweisungsoperator "->„ angesprochen. Ist dieses Element wiederum eine Structure, so werden deren Elemente mittels des Operators Punkt (.) angesprochen.

Zeiger zählen Zahlen

Eine besondere Art von Variablen sind die Zeiger in C. Sie

```
                                        OpenLibrary() geholt */
if(( wind=(struct Window *)OpenWindow(&nw))==NULL) exit();
/*   oeffnet Fenster
 *   falls kein Fehler vorliegt
    sonst Ausstieg*/
/* geschlossen wird das Fenster mit * CloseWindow(wind);
   ....*/
]
```

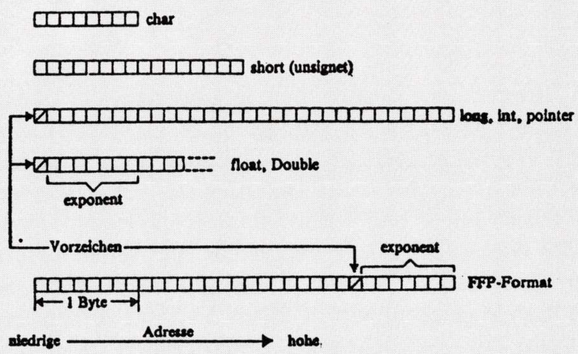
enthalten die Adresse einer anderen Variablen:

```
*zeiger ;   *vp ; *newwindow ; ...
```

Die Variable zeiger zeigt auf die Adresse von *zeiger, enthält also die Adresse in der der Wert von *zeiger steht.

```
int *zeiger; int x,y;
*zeiger = x  ; weist der Variabel *zeiger den wert von x zu
zeiger  = &x ; tut dasselbe, *zeiger weist auf Adresse x
zeiger  = x  ; nun zeigt *zeiger auf Adresse die in x stand
```

Der Operator & dient dem Errechnen der Adresse einer beliebigen Variablen.



Dieses Schaubild verdeutlicht den Speicherbedarf der einzelnen Variablen

Zeiger erlauben die Arithmetik mit Adressen und werden daher oft für speicherinterne Probleme eingesetzt. Das folgende Beispiel zeigt das Kopieren eines Vektors mit Hilfe von Zeigern:

```
int feld[100][100];
int *zeiger1,*zeiger2;

zeiger1 = &feld[0][0];          /* Anfangsadresse      */
zeiger2++ = zeiger1++           /* Wortweise kopieren  */
```

Zeiger sind sehr nützliche Werkzeuge, können aber Speicherbereiche des Betriebssystems angreifen. Deshalb sei zur Vorsicht bei der Benuztung von Zeigern (englisch: pointer) geraten! Zeiger müssen aber nicht unbedingt auf eine Variable oder ein Feld zeigen, sondern sie können auch auf Strukturen, Unions oder Funktionen zeigen. Nützlich ist diese Methode bei der Datenübergabe oder bei einem indirekten Aufruf. Dabei genügt es, den Zeiger als Referenz für einen Datenblock oder eine Funktion zu übernehmen, um nicht alle Daten durch den Übergabe-Engpaß schleusen zu müssen. Besonders wichtig sind Zeiger auf Funktionen, wenn, wie auf 32-Bit-Rechnern, die Datentypen von Zeigern und Funktionen nicht übereinstimmen.

Funktionen mit wichtigen Funktionen

Funktionen übernehmen die Aufgaben, die in anderen Programmiersprachen Unterprogramme ausführen. Eine Funktion erkennt man immer an den runden Klammern zum Beispiel bei

```
main(), for(i=1;i=10;i++),
oder
OpenWindow(),...
```

Funktionen können rekursiv sein, das heißt sie können sich selbst aufrufen. Sie geben falls nötig einen oder mehrere Werte zurück wenn am Ende der Funktion return() aufgerufen wird.

```
                                     OpenLibrary() geholt */
 if(( wind=(struct Window *)OpenWindow(&nw))==NULL) exit();
 /*   oeffnet Fenster
  *   falls kein Fehler vorliegt
     sonst Ausstieg*/
 /* geschlossen wird das Fenster mit * CloseWindow(wind);
    ....*/
 |
printf("Hello world \n");
```

ist das erste C-Programm das wir uns ansehen wollen. Der Ausdruck *main()* ist die Eintrittskarte zu jedem C-Programm. *main* selbst ist eine Funktion die auch Werte enthalen kann: *main(argc,argv)* enthält Aufrufparameter beim Start aus dem CLI. Die nach rechts offene geschweifte Klammer zeigt den Beginn einer Funktion oder eines Programms und die nach links offene das Ende.

```
/* Kommentar zum folgenden Programm ...                *
 * entnimmt Argumente aus der Kommandozeile(CLI)     . */

main(argc,argv)                        /* Argumentübergabe */
|
int argc,i;                            /* Argumente */
char *argv; BYTE z;                    /* Zeichenvariabel
                                        bereitstellen */

if(argc>0)                             /* Laufvariabel  */
for(i=0;i<argc;i++)
printf("Argument %2ld=%s\n",i,argv'i');
                                       /* Gibt Argumente aus */
else exit();
```

Dieses Beispielprogramm ersetzt das Kommando ECHO aus dem C-Verzeichnis der Workbenchdiskette. Mit der Angabe *main(argc,argv)* erkennt das C-Programm, daß es die Argumente, die beim Aufruf des Programms eingegeben wurden, übernimmt. So bei dem Aufruf > *echo "hans„* <. Das Programm gibt "hans„ auf dem Bildschirm aus. Die Funktion *prinf()* ist eine Standardfunktion, die wie alle anderen auf jedem C-Compiler verfügbar sind. *printf()* gibt eine Zeichenkette auf dem Bildschirm aus. Funktionen beherrschen den Character der Sprache C, und so werden auch Betriebssystemroutinen als Funktionen aufgerufen. So ist es zum Beispiel einfach ein Fenster zu öffnen:

```
#include <exec/types.h>              /* Headerfile mit
                                        Definitionen */
#include <intuition/intuition.h>     /* alle Intuition-
                                        Definitionen */
struct IntuitionBase *ib             /* Zeiger auf
                                        IntuitionLibrary */
struct Window *wind
struct NewWindow *nw = |             /* Zeiger auf neues
                                        Fenster */
20,                                  /* linke Ecke */
20,                                  /* obere Ecke */
300,                                 /* Breite     */
100,                                 /* Hoehe      */
0.                                   /* Farbe 1    */
1.                                   /* Farbe 2    */
"AMIGA-WELT C-Kurs".                 /* Titel      */
SMART_REFRESH | ACTIVATE,            /* Fenster-Modi*/
NULL,                                /* IDCMP Flag */
WBENCHSCREEN,                        /* Screen     */
NULL,NULL,NULL,NULL,                 /* weitere Flags*/
0,0,0,0
|,
main()
ib=(struct IntuitionBase
   *)OpenLibrary("intuition.library",0);
if(ib==NULL) exit();
                                     /* Zeiger auf Library mit
                                        OpenLibrary() geholt */
if(( wind=(struct Window *)OpenWindow(&nw))==NULL) exit();
/*   oeffnet Fenster
 *   falls kein Fehler vorliegt
    sonst Ausstieg*/
/* geschlossen wird das Fenster mit * CloseWindow(wind);
   ....*/
```

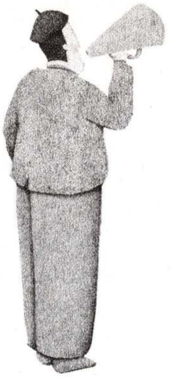
An diesem Beispiel kann man sehen, wie leicht Betriebssystemzugriffe in C realisiert werden können. Funktionen werden meistens zusammen mit Parametern aufgerufen, zum Beispiel *CloseWindow(wind)*, aber sie können auch ohne Parameter, wie in exit() aufgerufen werden. Wichtig ist bei der Parameterübergabe, daß die richtigen Werte bereitgestellt werden. Wenn eine Funktion eine *float*-Variable benötigt wird, muß sie auch bereitstehen, da sonst fatale Fehler auftreten können.

Wir werden in einer der nächsten Ausgaben eine Liste mit den wichtigsten Betriebssystemfunktionen abdrucken. Im zweiten Teil werden wir uns mit den Compilern und Linkern beschäftigen. Es sei jedoch empfohlen, bis dahin die Kenntnisse in C etwas aufzufrischen.

Liste der reservierten Wörter in C

auto	break	case	char	continue
default	do	double	else	entry
enum	extern	float	for	goto if
int	long	register	return short	
sizeof	static	struct	switch typedef	
union	unsigned	void	while	

(Thomas Kestler)



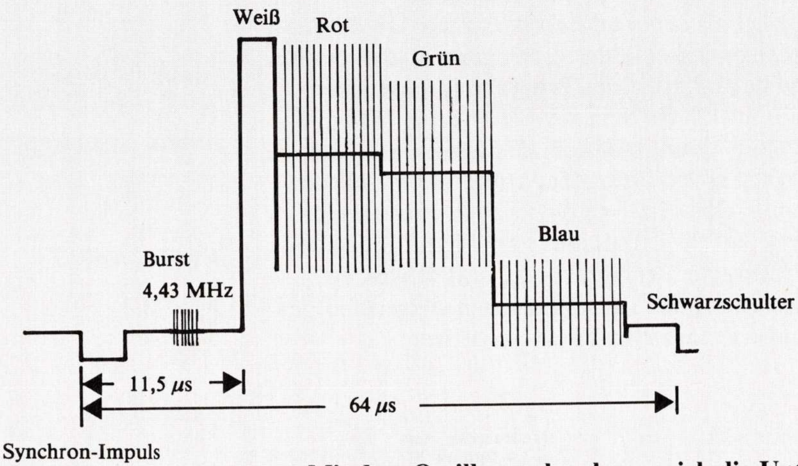
PAL – oder wie Farbe auf den Bildschirm kommt

Als Videocomputer hat sich der Amiga mittlerweile etabliert. Selbst ohne kostspielige Zusatzgeräte wie Genlocks und Digitizer bietet er gute Anwendungsmöglichkeiten im Videobereich. Daher ist es wichtig zu wissen, wie die PAL-Fernsehnorm definiert ist.

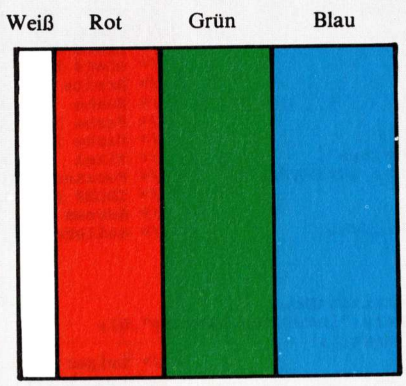
Die in der Bundesrepublik verkauften Amigas, die sogenannten PAL-Amigas, wurden von Commodore speziell dem „PAL“-Markt angepaßt. Sie erreichen eine maximale Auflösung von 640×512 Punkten im Vergleich zu den NTSC-Amigas mit einer maximalen Auflösung von 640×400 Punkten. Allerdings haben die PAL-Amigas den Nachteil, daß sie mit einer Bildfrequenz von 50 Hertz arbeiten. Im Gegensatz dazu arbeiten die übrigen Amigas mit einer Bildwiederholfrequenz von 60 Hertz. Dies macht sich besonders negativ beim Interlaced-Modus bemerkbar, dort verringert sich die Bildwiederholfrequenz auf die Hälfte, also auf 25 Hertz, während die Bilder der amerikanischen Versionen 30mal pro Sekunde aufgefrischt werden. Diese Unterschiede sind im unterschiedlichen Aufbau der Fernsehnormen begründet. Somit wurde gewährleistet, daß sich jeder Amigabesitzer ohne große Mühen ein Videosignal am Amiga nutzbar machen kann.

Allerdings haben nur die Besitzer eines Amiga 1000 die Möglichkeit, serienmäßig ein PAL-Farbvideosignal zu nutzen. Die Besitzer der neueren Amiga-Modelle können nur mit Hilfe einer PAL-Videokarte (Amiga 2000) oder eines Modulators A 520 (Amiga 500) an dieses Farbsignal gelangen.

Nach der Einführung des Schwarzweiß-Fernsehens stellte sich in den 50er Jahren die Frage nach der Norm für das zukünftige Farbfernsehen. Hierbei waren besondere Anforderungen zu berücksichtigen. Zum einen mußten alle Schwarzweiß-Fernsehbesitzer zukünftige Farbsendungen auf ihren Geräten empfangen können. Gleichzeitig mußten die neuen Farbfernseher in der Lage sein, Schwarzweiß-Sendungen zu empfangen. Aufgrund des Entwicklungsvorsprunges des Fernsehens in den Vereinigten Staaten wurde in der Zeit von 1951 bis 1953 von der NTSC (National Television



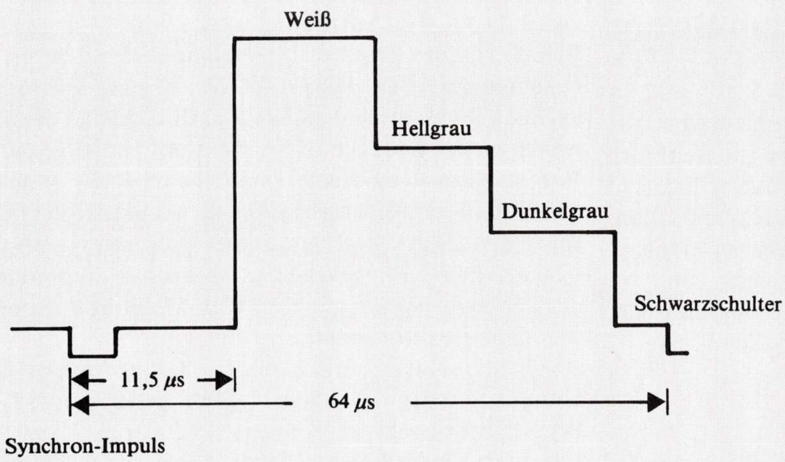
Mit dem Oszillographen lassen sich die Unterschiede zwischen Farb- . . .



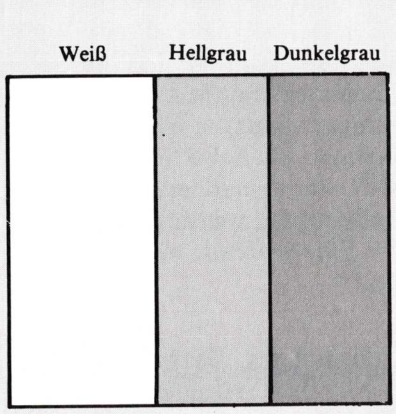
Weiß-, Rot-, Grün- und Blau-Daten ersetzten beim Farbsignal . . .

System Committee) eine Norm für das Farbfernsehen entworfen und anschließend festgelegt. Nach Ansicht führender Forscher beinhaltete diese Norm allerdings noch Mängel, die bei der Übertragung des NTSC-Signals unter ungünstigen Bedingungen die Farbwiedergabe verfälschten Daher wird das Kürzel NTSC des öfteren auch mit „Never the same Color" (niemals dieselbe Farbe) übersetzt. Professor Bruch, Mitarbeiter der Firma Telefunken, entwickelte daher auf Basis der NTSC-Norm die Farbnorm-PAL (Phase Alternating Line). Farbverfälschungen, wie sie bei Übertragungen des NTSC-Signals auftreten, sind bei dieser Norm nicht mehr vorhanden. Da in der Bundesrepublik noch keine Farbfernsehnorm eingeführt war, konnten Politiker und Fernsehmacher die bessere PAL-Norm für Deutschland verbindlich festschreiben. Erste Farbfernsehsendungen in PAL wurden in der Bundesrepublik 1967 ausgestrahlt. Neben den Normen PAL und NTSC gibt es noch die SECAM-Norm, die hauptsächlich in Frankreich, den ehemals französischen Kolonien und in der DDR eine Rolle spielt. Technisch gesehen gestaltet sich der Unterschied zwischen Schwarzweiß- und Farbsignal so, daß beim Farbsignal zusätzliche Informationen über den Farbinhalt des Bildes enthalten sind. Ein Wiedergabegerät erkennt an dem Vorhandensein des Color-Burst-Signals (Schaubild 2), daß ein Farbsignal vorliegt. Gleichzeitig ist das Color-Burst-Signal noch ein Synchronisationsimpuls für die in der Bildinformation zusätzlich enthaltenen Farbimpulse. Beim PAL-Signal wird mittels des Color-Bursts zusätzlich die Phasenverschiebung des Farbartzeigers angezeigt. Der Color-Burst hat eine Frequenz von 4,43 MHz und eine Länge von 12 bis 14 Perioden. Er folgt sofort nach dem Synchronisationssignal. Die eigentliche Farbbestimmung des Bildes erfolgt durch die dem Helligkeitsanteil (Luminanzsignal) überlagerte Farbinformation (Chromsignal). Der Unterschied wird deutlich, wenn Sie die „Treppenstufen" von Schaubild 1 und 2 vergleichen. In Schaubild 1 erkennt man saubere Treppenstufen, die Höhe einer Treppenstufe bestimmt die Helligkeit in diesem Bildbereich. Bei einem Farbbild wird durch zusätzliche Schwingungen an dieser Stelle die Farbinformation übermittelt. Die in den Schaubildern dargestellte Bildinformation ist immer für eine Zeile verantwortlich. Eine Zeile besteht aus 833 Bildpunkten, das heißt pro Bild ergibt sich eine Auflösung von 520 625 Bildpunkten. Ein Farbbild setzt sich aus 625 Zeilen zusammen, pro Sekunde werden 25 Bilder übertragen, somit ergibt sich eine Zeilenfrequenz von 15 625 Hz. Eine Zeile wird somit in 64 Microsec. übertragen. Pro Sekunde werden 13 015 625 Bildpunkte übertragen, dies entspricht einer Punktfrequenz von 13 MHz. Um ein Flackern des Bildes zu vermeiden, wird ein Bild in sogenannten Halbbildern übertragen. Im ersten Durchgang werden die Zeilen mit ungeraden Zeilennummern übertragen, im zweiten anschließend die Zeilen mit geraden Zeilennummern. Somit werden pro Sekunde 50 Halbbilder übertragen, es ergibt sich eine Bildwechselfrequenz von 50 Hertz, Bilder erscheinen so flackerfrei.

(Stefan Jäger)



. . . und Schwarzweißsignalen deutlich feststellen.



. . . die Weiß- und Grauwertanteile.

Einstieg mit BASIC

2. Teil: Einige Runden in Schleifen drehen

Willkommen zum zweiten Teil der Serie über Programmieren mit AmigaBASIC. Nach den ersten Schritten in der AMIGA WELT 4/88 dreht sich jetzt alles um Schleifen.

Schwimmen mit dem Strom

In Basic wird ein Programm vom Anfang (beginnend mit der ersten Zeile, eine Programmzeile nach der anderen) bis zum Ende verarbeitet. Dafür gibt es im Computer-Englisch die treffende Bezeichnung „top-to-bottom execution". Man kann das auch den Kontrollfluß des Programmes nennen, so als ob die Zeile, welche gerade ausgeführt wird, den Computer kontrollieren würde. Wurde eine Programmzeile abgearbeitet, so übergibt sie die Kontrolle an die nächste Zeile — jedoch nicht immer.

Schleifen-Raserei

Nehmen wir an, Sie müssen ein Programm schreiben, welches von der Tastatur eingegebene Zahlenpaare addiert und das Resultat ausgibt. Das ist kein schwieriges Programm; zwei INPUT- Anweisungen und eine PRINT-Anweisung genügen dazu. Nun nehmen wir zusätzlich an, daß zehn Zahlenpaare addiert und die Teilsummen ausgegeben werden sollen. Müssen Sie dazu zwanzig INPUT- und zehn PRINT-Anweisungen in das Listing eintippen? Nein, denn das Programm kann das (viel weniger aufwendig) in einer sogenannten Schleife für Sie erledigen.
REM Schleifen-Demo (1)

```
Start:
 INPUT "1.Zahl";z1
 INPUT "2.Zahl";z2
 PRINT z1+z2
GOTO Start
END
```

Ohne die GOTO-Anweisung würde das Programm zwei Zahlen von der Tastatur übernehmen, sie addieren und das Ergebnis ausgeben. Die GOTO-Anweisung aber ändert den in Block 9 beschriebenen Kontrollfluß, indem die Kontrolle wieder an die Zeile die „Start:" enthält zurückgegeben wird. Von dort aus wird wieder eine Zeile nach der anderen bis zur GOTO-Anweisung bearbeitet, wo wieder der Rücksprung zu „Start:" erfolgt. Dieses Programm erreicht daher niemals die END-Anweisung. Abbrechen kann man es nur mit der Stop-Option aus dem Run-Menü oder dem entsprechendem Tastaturbefehl.
Das Schlüsselwort „Start" werden Sie im AmigaBASIC-Handbuch vergeblich suchen, denn dieses Wort ist kein BASIC-Schlüsselwort, sondern wird hier als Label (Markierung) für den Beginn der Schleife verwendet. Damit dieses Label von BASIC nicht als Variable interpretiert wird, muß es mit einem Doppelpunkt abgeschlossen werden. Der Name eines Labels ist frei wählbar und statt „Start:" könnte jede andere Bezeichnug gewählt werden. Im Gegensatz zu anderen BASIC-Dialekten braucht AmigaBASIC keine Zeilennummern. Sprunganweisungen wie GOTO brauchen aber einen Bezugspunkt. Ein Label liefert diesen. Das Label „Start:" und „GOTO Start" markieren den Anfang und das Ende einer Kontrollstruktur, welche den normalen Programmfluß ändert. Eine Struktur, wie im Beispiel, nennt man „Endlosschleife".

Entscheidungen, Entscheidungen...

Es gibt nur wenig Anwendungsfälle für Endlosschleifen. Wesentlich nützlichere Kontrollstrukturen sind solche, in denen festgelegt wird, wie oft ein Programm eine Schleife ▶

Wir sind ein Mitglied der International Data Group, der Welt größter Verleger für computerbezogene Informationen. Die Gruppe veröffentlicht 90 Computer-Publikationen in 33 Ländern. 14 Millionen Menschen lesen eine oder mehrere Publikationen dieser Gruppe pro Monat. Die Mitglieder sind am CW-Communications International News Service angeschlossen, einem täglichen Nachrichtendienst für die aktuellsten Meldungen aus dem internationalen DV-Geschehen.

Wir suchen für

MS-DOS WELT

AMIGA WELT

Programmautoren, Testexperten, freie Mitarbeiter, Bastler.

Sie sollten Erfahrung im Umgang mit Computern, aber auch Kenntnisse der zugehörigen Hard- und Software haben. Kontaktfreude und Grundkenntnisse der gängigsten Programme wie Textverarbeitung und Grafik setzen wir voraus. Die Honorierung entspricht der,vergleichbarer Verlage. Eventuell anfallende Reisekosten werden ersetzt. Eine Textprobe von einem möglicherweise bereits veröffentlichten Beitrag würde uns natürlich sehr freuen. Für einen ersten Kontakt steht Ihnen zur Verfügung:

Siggi Pöschel
Tel. 0 89/3 60 86-210
CW-Publikationen Verlagsgesellschaft mbH
Rheinstr. 28
8000 München 40

durchläuft und unter welchen Bedingungen sie verlassen wird. Solche Strukturen sind im AmigaBASIC der Normalfall, weil durch sie das Programm die Fähigkeit erhält, Entscheidungen zu treffen. Ändern Sie das Programm aus dem vorherigem Kapitel wie folgt...
REM Schleifen-Demo (2)

```
Start:
 INPUT "1.Zahl";z1
 INPUT "2.Zahl";z2
 PRINT z1+z2
IF y <> 999 GOTO Start
END
```

...und lassen Sie das Programm laufen. Wenn es zur vorletzten Zeile kommt, testet es, ob der Wert von y ungleich 999 ist. Bei jedem Wert, der von 999 verschieden ist, springt das Programm zum Label „Start:" zurück. Hat y den Wert 999, dann wird der Sprung nicht ausgeführt und das Programm wird mit END beendet. Mit Hilfe der IF...GOTO-Anweisung trifft das Programm eine Entscheidung. Damit die Sache für Sie nicht zu verwirrend wird, ist jetzt eine ausführliche Erklärung fällig:
Die IF-Anweisung erlaubt die Verzweigung in verschiedene Programmabschnitte oder die Ausführung verschiedener Anweisungsblöcke, abhängig vom Wahrheitsgehalt eines logisch auswertbaren Ausdruckes. Ein Ausdruck (der in der Regel eine Bedingung enthält) ist „wahr", wenn sein Wert von Null verschieden ist; logisch „falsch", wenn sein Wert Null ist. Wir werden uns im Rahmen dieser Serie noch näher mit den logischen Operatoren befassen. Wenn Sie jetzt schon mehr darüber wissen wollen — im AmigaBASIC-Handbuch stehen die Grundbegriffe (Kapitel 8.6).
Die IF-Anweisung kennt mehrere Formate:
1. „wahr" oder „falsch":

```
IF Ausdruck GOTO Label
```

Wenn der Ausdruck wahr ist, findet ein Sprung zu dem in GOTO definiertem Label statt.

```
IF Ausdruck THEN Anweisung
```

Wenn der Ausdruck wahr ist, wird die Anweisung hinter THEN ausgeführt.

```
IF Ausdruck THEN Anweisung1 ELSE Anweisung2
```

Wenn der Ausdruck wahr ist, wird Anweisung1 ausgeführt. Ist er falsch, so wird Anweisung2 ausgeführt. In dieser Form darf die IF...THEN-Anweisung maximal 255 Zeichen lang werden. Für eine umfangreiche Kontrollstruktur ist das manchmal zu wenig.

3. AmigaBASIC kennt daher eine weitere Form der IF...-THEN-Anweisung, bei der zwischen THEN und ELSE beliebig viele Anweisungen stehen dürfen:

```
IF Ausdruck1 THEN
 Zeilen mit Anweisungen
ELSEIF Ausdruck2 THEN
 Zeilen mit Anweisungen
ELSEIF Ausdruck3 THEN
 Zeilen mit Anweisungen
ELSE
Zeilen mit Anweisungen
ENDIF
```

In dieser Kontrollstruktur werden mehrerer Ausdrücke überprüft. Ist einer davon wahr, werden die Befehle in den folgenden Zeilen ausgeführt, bis ein ELSE, ELSEIF oder ENDIF auftaucht. Dann wird hinter die ENDIF-Anweisung gesprungen. Ist keiner der Ausdrücke wahr, werden hingegen die Befehle zwischen ELSE und END IF ausgeführt. Jede IF-Struktur muß mit END IF abgeschlossen werden. Geschieht das nicht, so meldet sich AmigaBASIC mit einer Fehlermeldung oder es kommt zu unerwünschten Reaktionen.
Achtung: ELSEIF muß zusammengeschrieben werden! Die im AmigaBASIC-Handbuch angegebene Syntax ist falsch! Wie kompliziert eine solche IF...THEN-Struktur aufgebaut sein kann, zeigt das Beispiel „IF...THEN Quiz". Studieren Sie dieses Listing und versuchen Sie herauszufinden, welchen Output dieses Programm liefert.

```
REM IF...THEN Quiz
Zahl1=5:Zahl2=7:Zahl3=10
IF Zahl1 + Zahl2 = Zahl3 THEN SprungMarke
IF Zahl1 + Zahl1 = Zahl3 THEN
  PRINT "Es war einmal "
  IF 2 * Zahl1 = Zahl2 THEN
    PRINT "ein kleines Programm"
  ELSE
   PRINT "eine IF...THEN Struktur,"
  END IF
  PRINT "die ziemlich kompliziert war."
ELSE
 IF Zahl3 - 3 = Zahl2 THEN
   PRINT "und weil sie niemand verstand"
   PRINT "lief sie unentwegt weiter"
 ELSE
   PRINT "sicherlich hatte sie einen Zweck"
 END IF
END IF
SprungMarke:
 IF Zahl3 = (2*(Zahl2 -Zahl1)) THEN
   PRINT "aber ihre Bedingungen"
   PRINT "wurden nie erfüllt"
 ELSE
  IF Zahl3 = (Zahl1 * Zahl2)-15 THEN
    PRINT "als sie ihr Ende fand"
  ELSE
   PRINT "Sie können diese aber nur verstehen,"
  END IF
 END IF
PRINT "wenn Sie wie ein Computer denken!"
END
```

Sicherlich haben Sie die ineinander verschachtelten „IF...END IF"- Strukturen durchschaut. Wenn nicht, hier ist der Output dieses Programmes:
Es war einmal eine IF...THEN Struktur, die ziemlich kompliziert war. Sie können diese aber nur verstehen, wenn Sie wie ein Computer denken!

Die Kontrolle bleibt beim Programmierer

AmigaBASIC kennt zwei primäre Schleifenstrukturen: FOR..NEXT und WHILE...WEND Schleifen.
FOR...NEXT-Schleifen ermöglichen die Mehrfachverarbeitung einer Folge von Anweisungen, Befehlen oder Funktionen in einer Schleife mit einer genau definierten Zahl von

Durchläufen. Sie können sogar mehrere FOR...NEXT-Schleifen ineinander schachteln.
Die Schleifenparameter werden mit der FOR-Anweisung festgelegt, die zum Beipiel folgendes Format haben kann: FOR x = 1 TO 100. Der Variablen x, die auch einen anderen Namen haben kann und als Lauf-Variable dient, wird der Startwert der Schleife zugewiesen. Hier wurde der Wert 1 gewählt. Es könnte aber auch eine Variable sein. Für den nach TO folgenden Endwert gilt das gleiche. Hier hat er den Wert 100.
Als nächstes folgt der sogenannte Schleifenkörper, in welchem die Zeilen mit den auszuführenden Anweisungen stehen. Abgeschlossen wird die Schleife mit einer NEXT-Anweisung.

```
FOR Z=5 TO 25
 quadratZ=Z*Z
 PRINT Z,quadratZ
NEXT Z
END
```

Dieses Programm gibt alle Zahlen und ihre Quadrate zwischen 5 und 25 aus. Es beginnt mit dem Startwert 5. Jedesmal wenn es die NEXT-Anweisung erreicht, wird Z um 1 inkrementiert und das Programm springt zur FOR-Anweisung zurück. Hier wird getestet ob Z schon den Wert 25 erreicht hat. Wenn nein, erfolgt der nächste Durchgang. Wenn ja, folgt noch ein Durchgang und das Programm wird mit der auf NEXT folgenden Anweisung fortgesetzt.
Die FOR-Anweisung verträgt auch krumme Werte, wie zum Beispiel:

```
FOR i=3.2 TO 46.8 STEP .2
```

In dieser Zeile ist eine weitere Option der FOR-Anweisung enthalten. Ohne STEP wird der Schleifenzähler, wie Sie bereits wissen, nach jedem Durchlauf um eins inkrementiert. Mit STEP wird ein vom Programmierer gewünschtes Inkrement festgelegt.

WHILE...WEND-Schleifen arbeiten nach einem anderen Algorithmus und haben folgenden prinzipiellen Aufbau:
WHILE Ausdruck Zeilen mit Anweisungen WEND
Trifft das Programm auf eine WEND-Anweisung, so wird geprüft, ob der Ausdruck wahr ist. Wenn ja, so werden alle Anweisungen bis zu WEND ausgeführt. Jedesmal wenn das Programm zu WEND kommt, springt es zu der zugehörigen WHILE-Anweisung zurück und überprüft diese erneut.
Das Spiel geht solange weiter als der Ausdruck wahr ist. Innerhalb der Schleife wird der Ausdruck (meist eine Variable) üblicherweise so verändert, daß der Ausdruck irgendwann einmal falsch wird. Geschieht das nicht, so wird das eine Endlosschleife.

```
Zaehler=0
Zahl=1
WHILE QuadratZahl < 1000
 QuadratZahl = Zahl * Zahl
 Zaehler = Zaehler + 1
 Zahl = Zahl + 1
WEND
PRINT "Die Anzahl aller
       Quadratzahlen unter 1000 ist";Zaehler
END
```

Im obigen Beispiel wurde mit GOTO eine Endlosschleife produziert. Mit WHILE...WEND geht das kurz und einfach:

```
WHILE 1:WEND
```

Hinter der WHILE-Anweisung steht zwar normalerweise eine logische Bedingung wie etwa „a < 10“. Nun hat aber AmigaBASIC die Eigenschaft allen, von Null verschiedenen Zahlen den logischen Wert „wahr“ zuzuordnen, während die Zahl Null logisch „falsch“ entspricht. Damit ist der Wert „1“ wahr und wir haben eine Endlosschleife programmiert.
(rw/iw) (wird fortgeführt)

Hollywood im Arbeitszimmer

Mit einem Computer kann man jedes Arbeitszimmer in ein Filmgelände verwandeln. Mit Know-how und bei entsprechendem Kleingeld lassen sich überraschende Effekte direkt vom Amiga aufs Videoband zaubern.

Video Desktop ist das neue Einsatzgebiet für Computer der 68000er-Klasse. Dinge, die bisher nur in hochaufgerüsteten Fernsehstudios realisierbar waren, können nun auch das private Videofilmen attraktiver machen. Schon mit der Standardhardware lassen sich interessante Effekte realisieren. Um jedoch Videoproduktionen zu erhalten, die höheren Ansprüchen genügen, sind noch einige Investitionen erforderlich.

Einstieg in das Thema verlangt Grundwissen

Um das Medium Video besser verstehen zu können, und um falschen Hoffnungen auf eine schnelle Filmkarriere vorzubeugen, müssen erst einmal die Fakten zum Thema Video auf den Tisch. Zunächst sollte man sich klarmachen, daß die Bilder auf einem RGB- oder Composite-Monitor in keiner Weise dem entsprechen, was man auf Videobändern vorfindet. Der in den Amiga 1000 eingebaute Encoder (der Schaltkreis, der das RGB-Bild des Computers in ein Composite-Bild für Video konvertiert) weist nur schwache Leistungen auf; in Amiga 500 und Amiga 2000 fehlt der Encoder ganz. Die einzige Möglichkeit, sich eine Szene anzuschauen, besteht darin, sie aufzuzeichnen und auf einem Monitor oder Fernsehgerät abzuspielen. Auch dann hängt das Ergebnis noch davon ab, welche der drei Video-Normen zugrunde liegt. Der niedrigste Standard, nämlich Consumer, wird vorwiegend von Amateuren mit ½-Zoll-Videobändern gewählt. Profis, die nicht für das Fernsehen arbeiten und ¾-Zoll-Videobänder einsetzen, arbeiten lieber mit dem Industrie-Standard. Der anspruchsvollste Standard, nämlich Broadcast, wird bei Fernsehproduktionen mit Ein-Zoll-Bändern verwendet.

Während Amiga-Grafiken auf Ein-Zoll-Bändern ein ausgezeichnetes und auf ¾-Zoll-Bändern ein gutes Resultat hergeben, läßt die Qualität bei ½-Zoll-Bändern zu wünschen übrig. Der Aufzeichnungsbereich ist bei Ein-Zoll-Bändern größer, wodurch sich größere Toleranzen für fehlerhafte Signale ergeben. Darüber hinaus können die hochtechnischen Geräte in den Fernsehstudios, wie zum Beispiel Hochleistungs-Encoder, Videokorrekturverstärker und Time-Base-Korrektoren (die Zeit- und Phasenfehler im Videosignal korrigieren) kleinere Abweichungen ausgleichen. Angehende Video-Produzenten haben keinen Zugriff auf derartige Gerätschaften. Für sie gilt es, auf einfache Weise Probleme beim Aufzeichnen von Grafiken aus dem Wege zu räumen. Das eklatanteste dieser Probleme ist das horizontal verschobene Video-Signal, das von den meisten Amigas erzeugt wird. Dies kann jedoch wegen der Schwierigkeiten bei der horizontalen Phasenlage (die insbesondere bei den frühen A-1000 auftreten), nicht vollständig eliminiert werden. Das Bild läßt sich zwar am Monitor über Preferences zentrieren, doch kann dies nicht die Einzelbilddarstellung über den gesamten Bildschirm (Overscan-Modus), den man beim Video vorfindet, ersetzen. Das Bild der Videokamera wird durch ein Genlock (ein Gerät, das das Eingangssignal mit dem des Computers synchronisiert) geleitet. Da es den Umweg über ein zwischengeschaltetes Gerät nimmt, ist das Zentrieren eines solchen Signals etwas kompliziert. Das Genlock von Commodore, A-1300, verfügt über eine Justiervorrichtung für die Horizontalzentrierung, doch läßt sich diese nicht weit genug nach links einstellen, um das Problem vollständig aus der Welt zu schaffen. Auch das Öffnen des Genlocks und die Änderung der Grobeinstellung schafft hier keine Abhilfe. Statt das Problem zu beheben, führt dies lediglich zu extremen Farbverzerrungen, und darüber hinaus entfallen eventuelle Garantieansprüche wegen des Herumbastelns am Gerät.

Farbe muß her

Egal, ob das Bild bestens zentriert ist oder man den Fehler als exzentrische Absicht des Video-Künstlers tarnt — der sicherste Weg zum Profi geht über die aufmerksame Beobachtung von Details und die Feinabstimmung der Bilder. Im professionellen Studiobereich werden die Video-Geräte nach Maßgabe eines Standard-Sets von Farbbalken geprüft und justiert, um gleichmäßige und satte Farben zu gewährleisten. So etwas läßt sich mit Deluxe Paint II selbst erzeugen. Hierzu teilt man den Bildschirm vertikal in acht Balken auf und stellt die Zeiger der Farbpalette entsprechend der folgenden Wertetabelle ein: ►



Illustriert von: Paul Yalowitz

	R	G	B
Weiß	13	13	13
Gelb	13	12	1
Cyan	0	13	10
Grün	2	12	0
Magenta	13	0	15
Rot	14	0	5
Blau	1	1	12
Schwarz	1	1	1

Um sicherzustellen, daß diese Einstellungen der Industrienorm entsprechen, kann man sie mit einem Wellenform-Monitor (der das elektronische Muster eines Video-Signals mißt) und einem Vektorskop (für die Farbangleichung) kontrollieren.
Wenn die Personen in den ersten Videos stark gebräunt erscheinen und Kleidung in schreienden Signalfarben tragen, so liegt das nicht an Ihrem Mangel an Geschick. Die Farbsättigung (Chroma) ist beim Amiga etwas erhöht eingestellt und nicht veränderbar; man kann jedoch die daraus resultierende Farbübersättigung auf ein Mindestmaß reduzieren. Bei ½-Zoll- und ¾-Zoll-Bändern erzielt man ein optimales Ergebnis, wenn die Farben zum Malen auf den Wert 13 oder niedriger eingestellt werden. In den Farbbalken weisen die Farben Magenta und Rot Werte über 13 auf, doch diese Farben wurden für den Studio-Standard gewählt, nicht für den Amiga. Wenn man sie auf ½- oder ¾-Zoll-Bänder aufzeichnet, zeigen sie sich sehr instabil. Einige Farben sind stabiler als andere: Schwarz, Weiß und alle Grauschattierungen zählen zu den äußerst stabilen Farben. Grüntöne zeigen sich ebenfalls recht beständig. Blautöne weisen eine gewisse Instabilität auf, Braun- und Gelbtöne in noch höherem Maße. Rot und Magenta sind extrem instabil. Wenn man nicht gerade Psycho-Effekte hervorrufen möchte, sollten für Hintergründe und größere Flächen stabile Farben gewählt werden.
Genausowenig wie man Punkte und Karos zusammen tragen sollte, sollte man beim Video komplementäre Farben (Rot und Cyan, Blau und Gelb, Grün und Magenta) nebeneinander verwenden, da sie ineinander verlaufen und sich gegenseitig aufheben. Diese Effekte sind bei senkrechten und feinen Linien häufiger zu beobachten als bei horizontalen und breiteren Linien.
Wenn Sie Zugang zu professionellen Video-Geräten haben, können Sie das Problem der Farbsättigung um ein weiteres verringern, indem Sie die Farbsättigung mit einem Videokorrekturverstärker oder einem professionellen ¾-Zoll-Videorecorder auf einen niedrigen Wert einstellen. Da nur die ausgehenden Signale justiert werden, benötigen Sie in diesem Fall einen zusätzlichen Rekorder für die Aufzeichnung des korrigierten Bildes.

Achtung Aufnahme!

Für die Übertragung vom Amiga-Signalen auf Videoband sollte man den Composite-Video-Ausgang des Amiga und die Eingangsbuchse des Videorecorders mit einem abgeschirmten Kabel verbinden. Da der Amiga 500 und der Amiga 2000 nur RGB-Ausgangsbuchsen haben, benötigt man einen externen Encoder für die Umwandlung des RGB-Signals in ein Farb-Composite-Signal. Um Interferenzen zu vermeiden, sollte man den Videorecorder so weit wie möglich vom Computer entfernt aufstellen (mindestens 1,20 m). Wenn man nun die Aufnahmetaste am Videorecorder drückt, wird der Amiga-Bildschirm auf Video überspielt. Soll die Qualität des Videos dem Standard der Fernsehstudios möglichst nahekommen, arbeitet man am besten im Interlace-Modus. Das echte Fernseh-Signal ist ein Interlace-Signal und das einzige, das von professionellen Editing-Systemen erkannt wird. Videorecorder für den Heimbereich und ¾-Zoll-Videorecorder zeichnen auch in anderen Auflösungen auf. Die Bilder lassen sich ebenfalls editieren, sofern man keinen Time-Base-Korrektor benutzt. Schalten Sie einfach den Frame-Servo-Switch am editierenden Recorder aus, wenn Ihnen das lästige Piepsen der Warnsignale auf die Nerven geht.
Für das Erzeugen von Grafiken zum Überspielen auf Video (Dubbing) bietet DeluxePaint II vier verschiedene Auflösungsstufen an, zwei davon Interlace: LoRes (320 × 200 Bildpunkte), MedRes (640 × 200), Interlace (320 × 400, Interlace mit 32 Farben) und HiRes (640 × 400, Interlace mit 16 Farben). Möchten Sie von der Workbench 1.1 in Lo- oder MedRes vorliegende Bilder in Interlace auf Video überspielen, laden Sie einfach das PublicDomain-Programm Setlace über CLI. Wenn das Flimmern des Interlace-Modus erscheint, kann man das Malprogramm starten und mit der Aufnahme beginnen.
Setlace kann nicht mit der Workbench-Version 1.2 ausgeführt werden; hier müssen die Bilder in den Interlace-Modus konvertiert werden. Laden Sie das Bild zunächst in LoRes ein und ändern Sie anschließend das Format auf Interlace. Das Bild wird daraufhin auf die Hälfte seiner normalen Größe reduziert. Nehmen Sie jeweils einen Ausschnitt als Brush und arbeiten Sie zur Wiederherstellung der korrekten Größe mit dem Brush-Menü (Double Vert aktivieren). Bringen Sie den Ausschnitt dann in die korrekte Position zurück.

Elektronisches Schnipseln geht nicht mit der Schere

Viele Fernsehkonsumenten stellen sich das Schneiden von Videofilmen als mühselige Arbeit vor, die in Dunkelkammern von Fachleuten ausgeführt wird, die in kleine Bildschirme schielen und ellenlange Bänder zerstückeln. Im Grunde ist dieser Vorgang weder mysteriös, noch besonders umständlich. Im Gegensatz zum Filmband wird Videoband nicht zerschnitten und zusammengefügt; es wird elektronisch editiert, indem jeweils eine Szene vom Originalband auf ein Master-Tape dupliziert wird. Eine Synchronspur (Control Track), auf der die Bildsynchronimpulse in Längsrichtung am Rand des Videobandes aufgezeichnet werden, reguliert die exakte Geschwindigkeit des Bandes sowie den Anfangspunkt für jedes produzierte Einzelbild.
Vor dem Aufzeichnen der Synchronspur auf das Master-Tape muß zuvor festgestellt werden, ob das Video mit dem Assembly- oder mit dem Insert-Schnitt editiert werden soll.

Beim Editieren mit der Assembly-Methode wird die Synchronspur mit jeder einzelnen Szene aufgezeichnet; infolgedessen müssen die Szenen in der Reihenfolge der Aufnahme editiert werden. Beim Editing mit Insert wird zunächst die Synchronspur aufgenommen, so daß an jeder beliebigen Stelle Szenen eingefügt werden können, ohne die Synchronspur weiter berücksichtigen zu müssen. Die meisten Anwender ziehen die Montage mit Insert vor, da die Synchronspur für das gesamte Programm gleichbleibend ist und nicht unterbrochen wird, was einen weicheren Szenenübergang gewährleistet. Hierbei muß jedoch das Master-Tape vor Beginn der Montage mit der Synchronspur bespielt werden. Zum Aufzeichnen einer Synchronspur, die mit Insert editiert werden soll, schließt man die Videokamera an den Recorder an und schaltet beides ein. Nehmen Sie die Schutzkappe für das Objektiv nicht ab und ziehen Sie das Mikrofon heraus, um die Aufnahme unerwünschter Bilder und Geräusche zu vermeiden. Drücken Sie auf Aufnahme und lassen Sie das Gerät bis zum Ende des Bandes laufen.

Das Editieren mit Insert setzt im allgemeinen eine hochwertige und kostspielige Ausrüstung voraus, unter anderem ist ein Editing-Controller erforderlich, der eine Art Spezialcomputer darstellt. Besitzen Sie jedoch zwei Videorecorder, können Sie auf einfache Weise mit der Assembly-Methode arbeiten, vorausgesetzt, ein Gerät verfügt über die Backspace-Editing-Funktion (fast alle Recorder sind mit dieser automatischen Rückspulvorrichtung ausgestattet). Nimmt man mit einem Gerät, das über die Backspace-Editing-Funktion verfügt, eine Szene auf und drückt die Pause-Taste, wird die Aufnahme gestoppt und das Band zwei Sekunden lang zurückgespult. Drückt man dann erneut die Pause-Taste, um die nächste Szene aufzunehmen, spielt der Recorder die letzten beiden Sekunden der vorhergehenden Szene ab, bevor er in den Aufnahme-Modus schaltet. So wird dem Band Zeit eingeräumt, die richtige Geschwindigkeit zu erreichen und die Synchronspur der ersten Szene festzuhalten, bevor die nächste Szene mit der Synchronspur aufgezeichnet wird. Steht nur ein Videorecorder mit Backspace-Editing zur Verfügung, kann man Computerszenen aus dem Amiga direkt editieren. Man sollte jedoch aufpassen, da viele Recorder automatisch auf Stop umschalten, wenn die Pause-Taste länger als fünf Minuten gedrückt ist. Dies dient vor allem dem Schutz der Videoköpfe.

(D.L. Richardson/S. Bröker/Luk)

GLOSSAR

ADO (AMPEX DIGITAL OPTICALS) — Ein Gerät zur Erzeugung diverser Bildübergänge und Spezialeffekte beim Video. Wird vornehmlich in Rundfunkstudios eingesetzt.
ANIMATION — Eine Methode zur Simulation von Bewegung. Trickfilmtechnik.
ANIM FORMAT — Ein Standardformat zum Komprimieren von Animationen für Echtzeit-Wiedergabe. Befindet sich noch in der Entwicklungsphase.
AUFLÖSUNG — Die Anzahl der Bildpunkte auf einem Video-Bildschirm in horizontaler und vertikaler Richtung.
COMPOSITE VIDEO-SIGNAL — Eine Kombination aus Chrominanz-, Luminanz- und Synchronsignal.
DIGITIZER — Ein Hardwaregerät zur Umwandlung der Bildsignale einer Videokamera oder eines Videorecorders in digitale Informationen.
DITHERING MODE — Ein Modus, in dem ein schneller Farbwechsel an den einzelnen Bildpunkten den Eindruck einer neuen Farbe enstehen läßt.
DOUBLE BUFFERING — Zum Erzeugen von Weichanimation wird mit zwei oder mehr getrennten Speichern gearbeitet. Der erste zeigt das Bild an, während der zweite bereits das nächste Bild lädt.
ENCODER — ein „Signalmischer", der zu den einzelnen Komponenten rot, grün und blau die horizontalen und vertikalen Synchronsignale hinzufügt, um Composite-Video-Signale zu erzeugen.
FRAME BUFFER — Ein Gerät zum Aufzeichnen, Anzeigen und Speichern von Videoeinzelbildern.
GENLOCK — Ein Zusatzgerät, das die Bildsignale einer externen Videoquelle (Videokamera, Videorecorder) mit den Bildsignalen des Computers synchronisiert und kombiniert.
HALFBRIGHT — ein Chip zur Erhöhung der Farbanzahl in LoRes von 32 auf 64. Ist in neuere Amiga-Computer eingebaut.
HAM MODUS — Abkürzung für Hold-and-Modify-Modus. Der HAM-Modus ermöglicht das gleichzeitige Anzeigen von 4096 Farben.
KEY SIGNAL — Ein Schwarzweißsignal mit grafischen Informationen.
KEYER — Eine Vorrichtung, die im allgemeinen Bestandteil des Genlock ist und Computergrafiken über Video lagert.
KOLORIEREN — Einfärben von Schwarzweißbildern in beliebigen Farbtönen.
KOPIERSCHUTZ — Eine Methode zum Schutz einer Diskette oder eines Programms vor unberechtigtem Erstellen von Kopien.
MATTE — ein Wort aus dem Französischen. Zwei Bilder werden zu einem neuen Bild kombiniert, wobei ein Bild als Maske dienst.
OVERSCAN — eine Funktion der Hardware und Software, die das Ausfüllen des gesamten Videobildschirms bewirkt.
PAINTBOX — ein Zusatzgerät zum Anfertigen von Zeichnungen für Videoanwendungen. Dabei wird mit der Maus oder einem Zeichenbrett gearbeitet.
RAY TRACING — eine Methode, bei der eine oder mehrere Lichtquellen auf einen Punkt gerichtet werden, während der Computer jeden „Strahl" bis zum Ende abtastet.
RS-170 A — Eine von der Videoindustrie festgelegte Norm, die als Richtlinie für einwandfreie Synchronisation von Videosignalen und ihren Komponenten dient.
SUBCARRIER PHASE — ein Regler für die Chrominanzträgerphase (Hilfsträgerphase), mit dem die Farbsättigung (Chroma) zwischen O und 360 Grad justiert werden kann. Etwa vergleichbar mit der Farbeinstellung an den meisten Fernsehgeräten.
TITELGENERATOR — Hardwaregerät für die Übertragung von Titeltext auf Video.

Computermalschule, Teil 5:

Strahlen machen elastisch

Von Joel Hagen



Nicht vom Lügen, sondern vom langen Sitzen vor dem Bildschirm kommt die lange Nase, meint Joel

Mit „RGB Hazard“ will uns Joel Hagen weismachen, daß intensive Bildschirmstrahlung die Haut unnatürlich elastisch macht. Der kreative Kalifornier beweist die Folgen am eigenen Leib: Die Nase wird zum Rüssel, der Mund zum Briefkasten.



Hier tritt Joel Hagen den . . .



. . . Beweis an. RGB-Strahlen der Monitore . . .

Als ich mich zum ersten Mal mit Video-Digitizern beschäftigte und feststellte, daß man digitalisierte Bilder in Malprogramme laden kann, kam mir die Idee, alltägliche Aufnahmen zu skurrilen Darstellungen zu deformieren. Grimassen und Fratzen hatten mich seit frühester Jugend fasziniert.

Und hochgezüchtete Computertechnologie ist genau das richtige Instrumentarium, diese geringgeschätzte Körperkunst zu verfeinern. Ein US-weiter Wettbewerb gab den Kick, die Idee zu verwirklichen. Der Veranstalter, die kalifornische „Badge Developers Group", stachelte die User mit großzügigen Preisen an, ihren Maschinen äußerste Leistung abzuringen. Ich beteiligte mich mit einem karikaturistischen Beitrag zu einem ernsten Thema. Mein Demo zeigt die verheerenden Folgen der RGB-Strahlung: Wer lange vor dem Bildschirm hockt, bekommt nämlich eine extrem dehnbare Gesichtshaut.

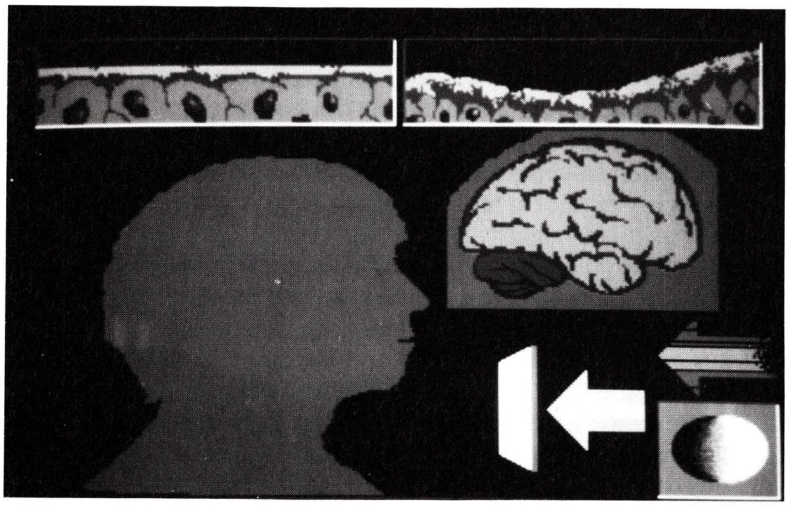
Um diese pathologische Veränderung glaubhaft darzustellen, brauchte ich ein Strahlenopfer. Was lag näher, als daß ich selbsterlittene Schäden auch an mir selbst demonstrierte? Schließlich hatte ich mich tagtäglich dem rot-grün-blauen Strahlenbombardement ausgesetzt. Vor dem gläsernen Auge meiner Schwarzweiß-Videokamera zog ich an meiner Nase, schob den Unterkiefer nach vorn und zerrte an meinen Mundwinkeln. Ein VHS-Videorekorder hielt alle mimischen Anstrengungen auf Band fest. Die so entstandenen IFF-Bilder wurden mit DigiView direkt auf Disk übertragen. Farbbilder bekommen beim Digitalisieren unansehnliche Muster. Dieser störende Effekt verschwindet, wenn man das Band über ein Fernsehgerät abspielt und mit einer Schwarzweiß-Kamera direkt vom Bildschirm filmt. Sofern DigiView mit einem Farbfilter versehen ist, lassen sich Farbbilder sogar direkt vom Band abnehmen.
Für den nächsten Schritt benutzte ich einen Videorekorder mit gutem Framefreezer. Damit wurde das bespielte Band Bild für Bild bearbeitet und mit Intervallen zwischen den herausgepickten Einzelbildern versehen; grundsätzlich ist dabei jedes gewünschte Intervall möglich. Über diesen Punkt herrscht Uneinigkeit. Die meisten User nehmen an, daß eine weiche Animation 30 bis 60 Bilder pro Sekunde erfordert. Zehn bis zwölf Bilder pro Sekunde ergeben jedoch einen fließenden Ablauf. Weiche Bewegung resultiert ja nicht nur, wenn alle Einzelbilder einer Videoszene verwendet wurden. Es genügt, jedes fünfte Bild herauszugreifen.
Um nun thematisch geeigneten Animationen einen wissenschaftlichen Touch zu geben, digitalisiere ich in schwarzweiß. Der dokumentarische Charakter wird verstärkt, wenn farbige Elemente dagegengesetzt werden. Außerdem lassen sich bei schwarzweißer Darstellung DPaints Tools SHADE und BLEND besonders effektvoll einsetzen. Diese „Werkzeuge" erlauben raffinierte Retuschen und Arrangements von Bildern. Im Demo zeichnete ich damit Mund und Nase in den Stadien extremer Verformung. Zuallererst mußte jedoch das wilde Farbendurcheinander der DigiView-Palette in eine Grauskala einsortiert werden. Diese Aufräumarbeit erledigte DeluxePaints Funktion SPREAD. Die Endpunkte null für Schwarz und 15 für Weiß definierten die Graupalette. Abschließend wurde die geordnete Palette zum RANGE erklärt, also zum Arbeitsbereich. Über die Option REMAP wurde das Bild mit der neuen Palette geladen. In dieser Form ließ sich das Bild beliebig aufspalten und wieder zusammensetzen. Das Werkzeug BLEND sorgte anschließend für fließende Übergänge. ▶



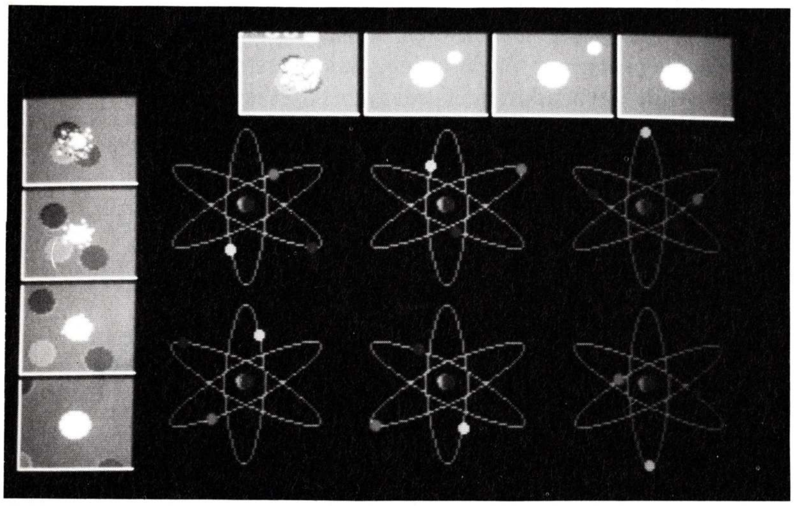
machen die Haut so elastisch, daß man . . .



. . . mit zwei Fingerbewegungen zum Breitmaulfrosch wird.



Die Elemente des Demos sind in IFF-Screens zusammengefaßt und werden . . .



. . . ebenso wie Animationssequenzen mit „The Director" über Blitteroperationen an die richtige Stelle projiziert

BLEND als auch SHADE eignen sich aber auch hervorragend zum Malen. Dabei ergeben sich Strukturen, die weniger einer Zeichnung als diffus verschmiertem Lehm ähneln. SHADE kann Umrisse in ihren Helligkeitsstufen variieren und so Raumtiefe und fotografische Effekte vorgaukeln. Jedes Einzelbild von meinem Gesicht unterzog ich einer solchen Behandlung.
Alle für das Demo benötigten Einzelbilder entstanden mit DigiView und DeluxePaint. Mit dem Animationsprogramm The Director wurde das Material aus stehenden Bildern zu einer lebendigen Sequenz. Ich hatte Director bereits für TV-Testsequenzen verwendet und zu meiner Lieblingssoftware für Sequenzen erkoren. Denn Director ist nicht nur ein Pageflipper und Effektgenerator, sondern bietet darüber hinaus eine sehr nützliche Grafik- und Animationssprache. Die Befehle sind denen des BASIC nahe verwandt. Wie in BASIC gibt es auch hier FOR/NEXT, GOSUB/RETURN und IF/ELSE/ENDIF-Bedingungen. Diese leistungsfähige Sprache eröffnet allen, die sich ernsthaft mit Grafik beschäftigen, ungeahnte Möglichkeiten. Mehrere Arbeitsschritte lassen sich gleichzeitig ausführen, ohne daß sich Farbenvielfalt oder Auflösung verringern. Zudem kann man die Leistung anderer Software ausnutzen, indem man sie mit Director kombiniert. So lassen sich aus den eingeklinkten Programmen Resultate herausholen, die sie ohne Director's Oberaufsicht nie zuwege gebracht hätten.
Für das RGB-Demo schöpfte ich Director's Möglichkeiten bei weitem nicht aus. Man könnte damit soviel anstellen. Doch fehlte mir bisher die Zeit, darin schlummernde Möglichkeiten zu erforschen. Die Spezialität des Programmes ist es, Amigas einzigartigen Blitter zu nutzen. Damit lassen sich jegliche Teile eines Screens animieren, ohne daß die nicht angesprochenen Abschnitte irgendwie davon betroffen wären. So wurde meine fertige Grimassenstudie in einen aktiven Kontext gesetzt und spielte dem Betrachter die kleine Geschichte als auch die visuellen Effekte vor. Es ereignen sich immer mehrere Dinge gleichzeitig. Damit jeder Bildschirm so aktiv und lebendig wurde, integrierte ich selbstentworfene Fonts in Director's Feature für Textgestaltung. Damit setzte ich den Begleittext in's jeweils aktuelle Bild ein. Die zwei kurzen „Atomic-Sequenzen" — aus sechs und acht Einzelbildern — entstanden mit DeluxePaint I. Da die Bildchen wenig Speicher brauchten, wurden sie zusammen mit anderen grafischen Kleinigkeiten auf zwei Screens geringer Auflösung gepackt. Auch der Hautquerschnitt wanderte in diese Sammlung bildnerischer Einheiten, die später an verschiedenen Punkten der Animation wieder gebraucht würden. Das Kommando BLIT erlaubt, definierte Teile aus allen IFF-Bildern herauszugreifen, die sich im Speicher befinden. Mit Hilfe des Blitters ließen sich diese Teilbilder dann — bei laufender Animation — in jeden beliebigen Bildpuffer kopieren. Gleichzeitig konnten IFF-Anim-Files, Farbveränderungen, Sounds und andere Effekte überwacht werden. Mit FOR/NEXT-Schleifen brachte ich das RGB-Kernspaltungsmodell auf seinem kleinen Screenausschnitt in Bewegung. Nach jeder Schleife bestimmte eine Reihe IF/ENDIFs, wann andere Ereignisse ausgelöst werden sollten, ob sich zum Beispiel das Gesicht auflösen (DISSOLVE) oder ein Pfeil bewegen sollte.
Die Animation war schnell fertig. Ich legte die Arbeit für einen Tag beiseite, um Distanz zu gewinnen. Erst, als ich meine Gedanken von „RGB" gelöst hatte, konnte ich Feinabstimmungen vornehmen. Directors Befehle PAUSE und SPEED halfen Zeitablauf und Bildfolgerhythmus zu optimieren. Nach sechs Tagen war „RGB Hazard" fertig. Ab damit! Auf dem Weg zur Post, kam mir meine Schulzeit in den Sinn. Damals hatte ich mit unterschiedlich bewertetem Erfolg Grimassen geschnitten. Heute — fast 30 Jahre später — habe ich diese Kunst mit Hilfe einer High-Tech-Ausstattung perfektioniert.

Joel Hagen gewann mit seinem „RGB"-Demo den ersten Preis beim Animationswettbewerb der „Badge Developers Group". In Deutschland war die humorvolle Sequenz erstmals auf der hannoverschen CeBIT zu sehen. Wer Joel schreiben möchte, kann seine Post (in Englisch) an die Amiga-Welt-Redaktion schicken.

500er Speichererweiterung

Für 512k zusätzliches RAM ● alle RAM's gesockelt ● selbstkonfigurierend ● abschaltbar ● ● Uhrenschaltung auf Platine mit Akku- bzw. Batteriepufferung nachrüstbar

Komplett mit 512k	Preis auf Anfrage
Superpreis mit Uhr	Preis auf Anfrage
Bauteilesatz für Uhr ohne Akku	24.-
Leerplatine mit Stecker	*39.-

*mit Schaltplan und Bestückunsliste

Profilaufwerk 3,5"

Metallgehäuse ● einstellbare Laufwerknummer mit Displayanzeige ● digitale Trackanzeige ● ● Write Protect am Laufwerk schaltbar ● abschaltbar ● durchgeschleifter Bus
1 Jahr Garantie

SuperALCOMPreis **329.-**

Trackanzeige

Für DF0-DF3 einstellbar ● für alle Laufwerke (3,5"/5,25") ● Laufwerkbus durchgeschleift ● mit Gehäuse

SuperALCOMPreis **69.-**

3,5" Laufwerk

Für alle Amiga's ● einstellbare Gerätenummer ● ● abschaltbar ● Metallgehäuse ● superflach ● 1 Zoll (2,54 cm) ● durchgeschleifter Bus ● TEAC Laufwerk
1 Jahr Garantie

komplett anschlußfertig	239.-
Amigafarbende Blende	+10.-

Basislaufwerke

1 Jahr Garantie

TEAC FD 135 FN 3,5" 1MB superslimline	218.-
TEAC FD 55 FR 5,25" 40/80 Tracks 1MB	239.-
TEAC FD 55GFR5,25" 40/80 Tracks	
Amigafarbende Blende	+10.-
1,6MB Diskchange	259.-
3,5" Gehäuse	25.-
5,25" Gehäuse	25.-
Gehäuse für "Gemischtes Doppel"	65.-

Laufwerkanschlußkabel

Zum Anschluß von Laufwerken an alle Amiga's ● ● mit Ansteuerelektronik

Für 3,5" Laufwerk	39.-
Für 5,25" Laufwerk	49.-

Steckplatzerweiterung 3-fach für Laufwerke

Jeder Steckplatz abschaltbar und einstellbare Laufwerksnummer ● Steckplatzerweiterung direkt am Amigagehäuse ● Dadurch keine Kabellängenprobleme

Anschlußfertig zum SuperALCOMPreis **49.-**



Laufwerk 5,25"

40/80 Track ● Laufwerksbus durchgeschleift ● abschaltbar ● einstellbare Adressen ● MS- DOS-kompatibel ● mit Diskchange

SuperALCOMPreis	298.-
HD 1,6 MB (umschaltbar)	318.-
Amigafarbende Blende	+10.-

Gemischtes Doppel 3,5/5,25"

einzeln ein-/abschaltbar ● einstellbare Laufwerksnummern mit Anzeige ● durchgeschleifter Bus ● bei 5,25" 40/80 Tracks umschaltbar ● Metallgehäuse ● 1 Jahr Garantie

SuperALCOMPreis **598.-**

Sampler Studio

● Professionelles Sampler-Programm ● 4-Kanal-Technik ● speichern auf 4 Disketten hintereinander möglich ● alle gängigen Formate (IFF, Data, Future) ● Echtzeitdisplay mit Zoomfunktion ● viele Verfremdungsmöglichkeiten ● Echo, Hall, Reverse **69.-**

Paket: Sampler + Software **129.-**

Soundsampler

Für alle Amiga's mit Software ● Type bei Bestellung bitte angeben ● 8-Bit Datenbreite ● Betrieb am Parallelport (Druckerport) ● Mit Vorverstärker für Micro-Anchluß (Cinch-Buchsen) ● Musik- und Sprachdigitalisierung möglich ● Arbeitet mit fast allen Digitizer-Programmen ● Formschönes Gehäuse

SuperALCOMPreis **79.-**

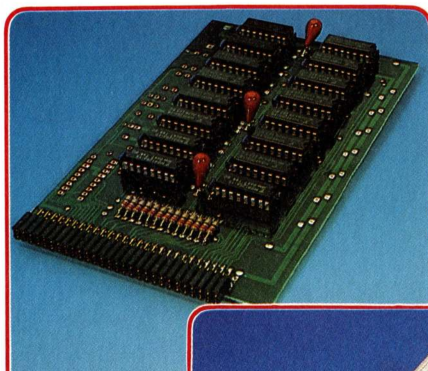
ausgereifte Ingenieurleistung ● 14 Tage Umtauschrecht ● 2 Jahre Garantie ● fast alle IC'S gesokkelt ● nur professionelle Leiterplatten ● Bauteile namhafter Hersteller ● mit Bedienungsanleitung

MIDI - Interface

4 Kanäle einschließlich 1 Thru ● Optische Datenanzeige ● Formschönes Gehäuse

Wahnsinnspreis von nur **89.-**

Bootselector **19.90**





ALCOMP
COMPUTERHARDWARE

Frischer Wind auf dem Hardwaresektor

Kickstartumschaltung

Bauen Sie die anderen Kickstart-Versionen in Ihren Amiga 500 ● Einfacher Einbau ohne Löten ● für Original-Kickstart-ROM und 2 zusätzliche Versionen auf EPROM ● EPROM-Programmierservice auf Anfrage

SuperALCOMPreis	59.-
Kickstartversion auf EPROM's	120.-

Userport + Experimentierkarte für Expansionport

Mit Lochraster und 2 x 6522 Ports

Leer	59.-
komplett aufgebaut	89.-

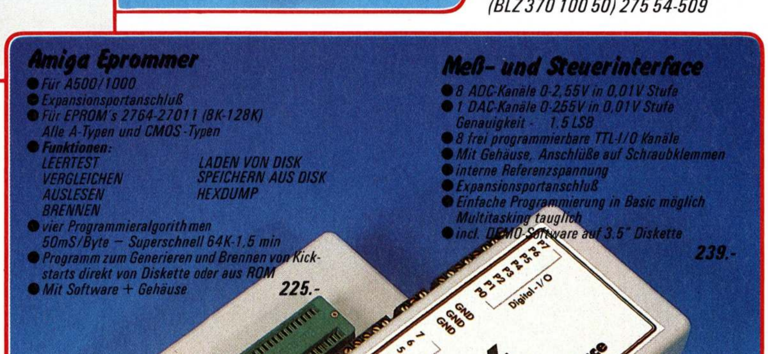
Wir suchen ständig Hardware-Entwicklungen. Wir garantieren gute Umsatzprovisionen und ehrliche Abrechnung

kostenloses Info anfordern!!!

Bestellung und Versand

ALCOMP
A. Lanfermann
Lessing Str. 46
5012 Bedburg
Tel. 0 22 72/15 80

Nachnahmeversand NN-Spesen 7.50 DM b. Vorkasse 3.- DM. Auslandsbestellungen: Nachnahmeversand NN-Spesen 10.- DM b. Vorkasse 5.-DM. Wir liefern Ihnen auf Ihre Rechnung und Gefahr zu den Verkaufs- und Lieferbedingungen des Elektronikgewerbes. Postgiroamt Köln (BLZ 370 100 50) 275 54-509



Entwurf + Druck PLUM, Düren-Oberzier, TEL. 0 24 28/7 11



Illustriert von José Cruz

Trimmplan für AmigaDOS

Mit einem strengen Trimm- und Diätplan namens ARP werden die CLI-Befehle aufgemöbelt und überflüssige Pfunde (sprich Bytes) abgebaut. Nun wird AmigaDOS so fit, daß mehr Verzeichnisse auf die Workbench passen und mehr Files ausgeführt werden können, als es mit den übergewichtigen AmigaDOS-Gegenstücken je möglich war.

Unter der Aufsicht der ARP-Trainer (AmigaDOS Replacement Project) werden die CLI-Befehle um überflüssige Pfunde erleichtert und kraftmäßig aufgepäppelt. Die AmigaDOS-Befehle, die ursprünglich in der schwerfälligen BCPL-Sprache (Basic Combined Programming Language, ein Vorläufer von C) programmiert wurden, sind langsam und umfangreich und besitzen nur wenig Power. Mit dem gemeinsamen ARP-Projekt unter der Leitung von Charlie Heath von der Firma Microsmiths hat man sich zum Ziel gesetzt, den erschlafften BCPL-Muskel zum Muskelpaket der 68000er Assembler-Sprache hochzutrainieren. Die ARP-Programme und auch die „arp.library", von der die Programme abhängig sind, sollen die bestehenden C-Directory-Befehle ersetzen. Die ARP-Befehle, die mit AmigaDOS-Befehlen völlig abwärtskompatibel sind, bieten zahlreiche Vorteile, unter anderem größere Globalzeichen-Fähigkeiten, eine erweiterte On-Line-Hilfe, residente Befehle, Umgebungs-Variablen, Ressource-Tracking und Verwalten von kleinen Speicherplatzmengen. Durchschnittlich verbrauchen sie nur halb so viele Kalorien wie die aktuellen AmigaDOS-Befehle — sie belegen auf der Workbench-Diskette etwa 35 KByte Speicherplatz und zeichnen sich dennoch durch größere Leistungsfähigkeit und höhere Verarbeitungsgeschwindigkeit sowie leichtere Zugänglichkeit aus.

Mit LOAD, SET, RUN das DOS auf Trab gebracht

Zu den augenfälligsten Verbesserungen des ARP gehören die neuen Befehle, für die keine BCPL-Pendants bestehen: ARUN, LOADLIB und SET. Zwei weitere Befehle, nämlich ASK und RESIDENT, sind für die Inhaber von AmigaDOS 1.2 neu, nicht jedoch für diejenigen, die bereits im Besitz der 1.3- Version sind. Wird ASK in Script-Files integriert, erlaubt es die Frage an den Anwender, ob Sonderaufgaben ausgeführt werden sollen. Mit RESIDENT können speicherresidente Befehle angelegt werden, die für eine noch schnellere Ausführung sorgen, als dies mit RAM-Befehlen möglich ist.

Der Befehl ARUN verbindet die Fähigkeiten von RUN mit denen des Public-Domain-Programmes RUNBACK. Wie auch das RUN von AmigaDOS kann ARUN ein Programm im Hintergrund ausführen und die CLI-Kontrolle an den Anwender zurückgeben. Mit den zusätzlichen Argumenten von ARUN lassen sich der Stack für das Programm sowie die Aufgabenpriorität festlegen. Bei verschiedenen Programmen kann man sogar das CLI-Fenster schließen, während das Programm im Hintergrund fortgesetzt wird, wodurch sich etwa 20 KByte Speicherplatz einsparen lassen. Ein geringfügiger Nachteil besteht darin, daß man Escape-Zeichen benutzen muß, um Strings mit ARUN ablaufen zu lassen. Die Sterntaste (*) ist das Standard-Escape-Zeichen des ARP; möchte man hierfür ein anderes Zeichen definieren, kann man dies mit Hilfe des neuen SET-Befehls realisieren. Anhand von LOADLIB lassen sich Libraries von Diskette oder Hard-Disk laden. Ist die Boot-Diskette bereits zu voll, um die gewünschten Libraries aufzunehmen, hat man die Möglichkeit, sie mit LOADLIB in das System zu laden. Mit SET lassen sich den Umgebungs-Variablen Werte zuordnen und so verschiedene Aspekte der Amiga-Systemumgebung vom Anwender konfigurieren. Ähnlich wie der SET-Befehl von Manx Aztec C, ist auch das SET des ARP mit den von Manx kompilierten Programmen, die Umgebungs-Variablen lesen können, kompatibel.

Zur Zeit unterstützt ARP verschiedene Umgebungs-Variablen, wie Copyflags, Dateformat und Listarchive. Mit Hilfe von SET und der Copyflag-Variablen kann man den COPY-Befehl anweisen, beim Anlegen von neuen Files bestimmte Eigenschaften der Ausgangs-Files zu übernehmen. Die Filenote, das Originaldatum und auch die Read-, Write-, Execute-, Delete- und Archive-Flags lassen sich duplizieren. Möchte man den SET-Befehl umgehen, kann man COPY anweisen, die File-Eigenschaften zu duplizieren, indem man das FLAGS-Keyword und seine Optionen in die COPY-Befehlszeile einbindet. Beide Methoden stellen eine Verbesse- ▶

rung gegenüber BCPL COPY dar, das Filenotes nicht zu duplizieren vermag und neue Files stets mit dem aktuellen Systemdatum versieht.
Die Dateformat-Variable definiert das Input- und Output-Format für Befehle wie DATE, LIST und SETDATE. Das Standardformat ist hierbei Tag-Monat-Jahr, doch mit SET und Dateformat läßt sich eine der vier möglichen Konstellationen auswählen.
Die Listarchive-Variable dient dazu, den LIST-Befehl anzuweisen, den Status eines File-Archive-Flags zusätzlich zu den RWED- Flags anzuzeigen. Erscheint bei den Flags ein A, deutet dies darauf hin, daß der File von einer Backup-Utility archiviert worden ist.

Ein Sternchen für jeden

Im Gegensatz zu der umständlichen und inkonsequenten Handhabung von Globalzeichen durch AmigaDOS, unterstützen die ARP-Befehle das Globalzeichen * auf ähnliche Weise wie MS-DOS. Fast jeder Befehl, von dem man annehmen könnte, daß er Globalzeichen-Argumente akzeptiert, ist dazu nunmehr imstande und erhöht damit die Leistung des CLI um ein Vielfaches. Mit dem neuen RENAME-Befehl lassen sich alle Befehle aus dem Directory ARP__c in das logische Device c: plazieren . Hierzu gibt man ein:
RENAME ARP__c/*to c

Vergleich von ARP mit den AmigaDos Befehlen

Befehl	ARP Bytes	DOS-1.3 Bytes	Zusätzliche Eigenschaften von ARP
ADDBUFFERS	405	876	Akzeptiert bis zu vier Drive/Buffer-Paare
ARUN	1176	NV	Kombiniertes RUN mit RUNBACK (pd); der Stack kann mit dem Stack-Schlüsselwort gesetzt werden; der NOIO Schalter erlaubt das Schließen des CLI obwohl ein Hintergrundtask läuft
ASK	556	648	Fragt bevor ein Batch-File ausgeführt wird
ASSIGN	968	2960	Akzeptiert mehrere Assign-Paare; — Option um mehrere Assign-Paare zu entfernen
AVAIL	NV	1772	NV
BINDDRIVERS	NV	2920	NV
BREAK	452	956	AK
CD	664	1756	Akzeptiert Wildcards; Root Schlüsselwort kann Laufwerken und Disketten zugewiesen werden
CHANGETASKPRI	488	1076	Vergleicht den Wert der Priorität
COPY	2244	9296	* = CON: nicht unterstüzt
DATE	1120	4208	AK
DELETE	1060	5568	ASK-Schalter mit verbesserten Wildcards
DIR	2120	8500	OPT H(Highlight directories), S(File sizes) oder HILITE, SIZE
DISKCHANGE	400	680	AK
DISKDOCTOR	NV	6672	NV
ECHO	260	564	AK
ED	NV	19564	NV
EDIT	NV	18164	NV
ELSE	540	860	AK
ENDCLI	NV	692	NV
ENDIF	40	40	AK
ENDSKIP	NV	40	NV
EXECUTE	NV	4532	NV
FAILAT	388	1028	AK
FAULT	NV	2688	(siehe ARP WHY-Befehl)
FF	NV	3200	NV
FILENOTE	356	690	AK
GETENV	NV	916	NV
IF	824	1600	AK
INFO	992	2068	Zeigt Speicherkapazität an
INSTALL	800	2416	AK
JOIN	560	1056	Verarbeitet und sortiert Filenamen mit Wildcards

Der RENAME-Befehl von BCPL gestattet jeweils nur die Umbenennung eines Files oder Verzeichnisses.
Neben der Unterstützung von Globalzeichen können ARP-Programme ebenfalls mehrere Argument-Sets manipulieren. So kann man mit einer einzigen ARP ASSIGN-Anweisung alle logischen Datenträger des Systems an die Festplatte übertragen. Unter AmigaDOS muß man ASSIGN für jedes Device erneut aufrufen, also insgesamt mindestens sieben separate Prozesse auslösen. Mit dem neuen ADD-BUFFERS kann man Buffer vier verschiedenen Drives auf einmal zuordnen, während MAKEDIR, RESIDENT und TYPE auch Mehrfach-Argumente akzeptieren.
Benötigt man eine Gedächtnisstütze für die neuen ARP-Befehle, gibt man einen Befehl und anschließend ein Fragezeichen ein. Wie bei den Original-Befehlen antwortet das System auch hier mit der Befehls-Schablone. Die Eingabe eines Fragezeichens hinter der Schablone bewirkt die Anzeige einer zweiten Zeile mit Erläuterungen.
Dem Durchschnitts-Anwender ist dies vielleicht weniger bewußt, doch das aktuelle Betriebssystem stellt hinsichtlich seiner Leistungsfähigkeit ein Handicap für den Programmierer dar. Er wird durch die DOS-Library und die wenig eindeutigen BCPL-Sprachkonventionen erheblich eingeschränkt und ist dazu gezwungen, mit zwei verschiedenen Konventionstypen umzugehen, einem, der von BCPL verstanden wird und einem weiteren für das restliche System.

Befehl	ARP Bytes	DOS-1.3 Bytes	Zusätzliche Eigenschaften von ARP
LAB	40	40	AK
LIST	2076	9696	SORT Option
LOADLIB	496	NV	Lädt eine Library von Disk
LOADWB	NV	2644	NV
LOCK	NV	2068	NV
MAKEDIR	416	768	Erstellt mehrere Directorys
MOUNT	2204	5432	Mountet mehrere Geräte; STARTPROC option lädt sofort den Driver oder Handler
NEWCLI	NV	2784	NV
NEWSHELL	NV	2732	NV
PATH	736	2136	Arbeitet mit Wildcards
PROMPT	484	584	AK
PROTECT	572	1396	Arbeitet mit Wildcards
QUIT	424	1036	AK
RELABEL	476	828	AK
REMRAD	NV	304	NV
RENAME	948	632	Unterstützt Wildcards
RESIDENT	1063	2620	AK
RUN	NV	2716	(siehe ARP ARUN)
SEARCH	1100	6332	Wildcards werden berücksichtigt
SET	572	NV	Definiert Environment Variablen
SETALERT	NV	3312	NV
SETCLOCK	NV	4556	NV
SETDATE	884	2632	Arbeitet mit Wildcards
SETENV	NV	836	(siehe ARP SET)
SKIP	664	1108	? option fragt den Anwender nach dem Sprunglabel
SORT	872	1868	AK
STACK	384	872	AK
STATUS	704	1752	AK
TYPE	1196	2284	Verarbeitet Wildcards, mehrere Filenamen; merged Files mit Wildcards in ein Destination File
VERSION	312	2424	Zeigt ARP-Version, Kickstart und Workbench-Version;
WAIT	664	1327	AK
WICH	NV	1872	NV
WHY	1376	576	Kombiniertes WHY und FAULT
XICON	NV	3156	NV

NV = nicht vorhanden
AK = abwärtskompatibel

▶

Mit der konsequenten „arp.library“ können Programmierer ihre Codes besser aufbauen und so effizienter Programme schreiben. Verschiedene Library-Routinen helfen dabei, den Dialog zwischen Programm und Anwender zu beschleunigen. Darüber hinaus lassen sich Umgebungs-Variablen definieren und in kommerzielle Software einbauen, um verschiedene Standard-Einstellungen zu ermöglichen. Die FileRequest()-Funktion von „arp.library“ stellt eine um ein Vielfaches verbesserte Version des Heath-File-Requesters dar, der als einer der schnellsten für den Amiga gilt. Die meisten Requester lesen das gesamte Verzeichnis und zeigen es an, bevor der Anwender einschreiten darf. Der Heath-Requester, der unter anderem in viele Aegis-Programme, in Photon Paint von MicroIllusions und in CityDesk von MicroSearch integriert wurde, erlaubt das Anklicken eines Files oder eines Verzeichnisses unmittelbar nach dem Erscheinen auf dem Bildschirm. Der Requester unterbricht sofort den aktuellen Verzeichnis-Lesevorgang und lädt entweder den File oder zeigt die Liste des neu ausgewählten Verzeichnisses an. Diese Verkürzung weiß man vor allem dann zu schätzen, wenn Verzeichnisse tief verzweigt sind . Ein weiterer Pluspunkt des ARP-Requesters gegenüber seinem Vorgänger liegt in dem verfügbaren „Parent Gadget“, das einen in der Verzeichnis-Hierarchie um eine Ebene zurückversetzt. Weiterhin gestattet es Programmierern, auf einfache Weise die verschiedensten Gadgets, wie DF0:, DH0:, und DH1: anzuhängen. Entwickler können ebenfalls String Gadgets (Felder für die Eingabe von Zeichen) einbauen, um Filter für eine selektive File-Anzeigean die Hand zu geben.

ARP-Requester gibt es auch mit Filter

Ein solcher Filter könnte bewirken, daß der Requester eines Malprogrammes nur Files mit dem Suffix „.pic“ anzeigt. Leitet der Anwender die DIR-Anzeige ein, wird das Window nicht mit irrelevanten Files überladen. Zur weiteren Beschleunigung des Zugriffs auf Verzeichnisse klickt man mit dem rechten Mausknopf in das Liste-Fenster. Der Requester zeigt die Liste aller verfügbaren physischen und logischen Datenträger an und läßt einen so direkt auf Files zuzugreifen, ohne daß man sich erst durch die verschiedenen Ebenen des Dateibaums quälen muß.

Weitere nützliche Funktionen wurden von den Entwicklern gleich miteingebastelt. Der GADS() Argument-Parser ist behilflich, wenn es darum geht, ein konsequentes Format für CLI-Befehle aufzustellen. Mit Assign() und ASyncRun() können Programme logische Datenträger zuordnen und externe Programme ausführen, ohne hierfür auf die Befehle ASSIGN und RUN zurückgreifen zu müssen.

Eine Funktion, die bei Betriebssystemen mit Multitasking eine große Rolle spielt, aber nicht im AmigaDOS enthalten ist, ist das „Resource Tracking“. Bei Multitasking-Prozessen müssen sich mehrere Programme die gleichen Ressourcen, nämlich Speicher, Files, logische Datenträger, teilen und diese Ressourcen wieder freigeben nachdem die Programme beendet sind, um nicht die Datenstruktur des anderen zu überschreiben oder Speicher zu blockieren.

Mit Hilfe der Funktionen in der „arp.library“ kann man Ressourcen automatisch nachvollziehen und durch Schließen wieder freigeben. Die Resource-Tracking-Funktion von ARP sorgt ebenfalls für geschicktes Verwalten von knapp gewordenem Speicherplatz, indem es in kritischen Situationen die Freigabe der Speicherplatzreserve eines Programmes bewirkt, vorausgesetzt, daß dieser Speicherplatz zu dem Zeitpunkt nicht vom Programm benötigt wird.

DIR erkennt jetzt Fileumfang

Abgesehen von den Neuerungen bei den Befehlen versieht AmigaDOS ARP-Staples mit besonderen Fähigkeiten. CD und PROMPT lassen sich für eine Anzeige des aktuellen Verzeichnisses, der Task-Nummer und eines Text-Strings in der CLI-Eingabezeile zusammen verwenden. Der ebenfalls neue Befehl CHANGETASKPRI ist imstande, die Priorität einer Aufgabe zu ändern, die bereits ausgeführt wird, und DELETE akzeptiert nun eine unbeschränkte Anzahl von Files als Argument; ein ASK-Keyword kann bei Löschungen mit Globalzeichen verwendet werden. Der DIR-Befehl bietet Optionen für die Anzeige der Dateigröße, für das ausschließliche Anzeigen von Files oder für die Unterscheidung von Verzeichnissen im normalen und hervorgehobenen Zustand. Die String-Argumente in ECHO kommen nun ohne Anführungszeichen aus, und dank seiner neu erworbenen Fähigkeit, Mehrfach-Argumente zu akzeptieren, läßt sich mit MAKEDIR ein Verzeichnis erstellen mit der Möglichkeit, verschiedene nachfolgende Verzeichnisebenen darin zu verschachteln. Und das alles durch einmaliges Ausführen des Befehls.

Die ARP-Dokumentation auf Diskette umfaßt die beiden Abschnitte UserDocs und ProDocs, wobei der letztere sich vornehmlich an Programmierer richtet. Sie ist nach den Konventionen abgefaßt, die auch im AmigaDos-Manual von Bantam Books berücksichtigt werden, wobei einige Beispiele und Beschreibungen jedoch einer Erklärung bedürfen. Der Beispiel-Befehl für die Verwendung von ARUN mit String-Argumenten beinhaltet ein Escape-Zeichen. Mit SET kann ein beliebiges Escape-Zeichen zugeordnet werden, doch das vorgegebene Zeichen ist *. Die COPY-Schablone im Handbuch bezeichnet FLAGS irrtümlicherweise als einen Schalter und nicht als ein erforderliches Keyword; die On-Line-Schablone weist FLAGS/K aus. Wenn man jedoch einen COPY-Befehl abgesehen vom Flags-Argument insgesamt richtig eingibt, zeigt COPY die erweiterte Help-Meldung an und nicht die Schablone. Das FLAGS-Keyword wird überhaupt nicht berücksichtigt. Das Integrieren der Befehle in das System ist relativ einfach. Der Execute-File auf der Diskette installiert ARP automatisch auf der Boot-Diskette. Arbeitet man lieber mit CLI, kann man folgende Befehle tippen:

copy ARPv1.1:libs/arp.library to libs:
copy ARPv1.1:c to c:
loadlib libs:arp.library
Folgende Befehle werden von der ARP-Version 1.1 zur Zeit nicht unterstützt:
BINDDRIVERS, DISKDOCTOR, ED, EDIT, ENDCLI, EXECUTE, FAULT (das WHY in ARP erfüllt die Aufgaben von WHY sowie auch von FAULT), LOADWB, NEWCLI und RUN (siehe ARUN in ARP).
Es wird jedoch hart an ARP gearbeitet, und sicherlich werden uns die Entwickler schon in Kürze mit Erweiterungen zum ARP Befehls- Set beglücken, die unter anderem auch das Fast-File-System der 1.3-Version von AmigaDOS unterstützen.

Die ersten kommerziellen Programme verfügen bereits über ARP, so zum Beispiel TxEd Plus von Microsmiths Inc. mit der arp.library. Der Sub-Battle-Simulator von Epyx bedient sich des ARP-File-Requesters. Auch die Programme Redact (ein DTP-Programm von Top own Development) und WShell (ein CLI-Enhancer von William S. Hawes) schließen sich der ARP-Parade an. Bei Commodore wurde man bereits auf dieses Projekt aufmerksam. Wie Heath mitteilte, haben die Programmierer die aktuelle ARP-Version Commodore kostenlos zur Verfügung gestellt. Angeblich zieht Commodore eine Implementation in Betracht, will sich aber wegen der noch nicht abgeschlossenen Verhandlungen zur Zeit nicht äußern.

(Jeffrey Blume)

Programmieren mit gespaltener Zunge

AmigaDos ist nicht aus einem einzigen Guß entstanden. Die ersten Amiga-Programmierer schrieben den Großteil des Amiga-ROM-Kerns in der effizienten 68000 Assemblersprache oder in C. Dann begann jedoch Metacomco, das DOS in BCPL zu programmieren.

Wichtige Teile des ROM-Kerns, wie die „intuition.library", die „dos.library" und die „graphics.library" sind in 68000-Assembler oder in C geschrieben. Mit den DOS-Implementationen in BCPL von Metacomco kam es zu Unstimmigkeiten, da sich die BCPL-Konventionen von den C-Konventionen und denen der 68000-Assemblersprache stark unterscheiden. Als Folge davon ließ sich DOS nicht einwandfrei in das restliche System integrieren. Im einzelnen ergeben sich Probleme aus der Art und Weise, wie BCPL Speicher adressiert (BPTRs), wie sie Strings (BSTRs) handhabt, sowie weiterhin im Bereich von Stacks.

Mit BPTR ist ausschließlich Longword-Alignment möglich. Um sicherzustellen, daß Speicheradressen ausgerichtet sind, teilt BPTRs die Adressen durch vier, bevor es auf die „dos.library" zugreift. In C kann der Compiler Adress-Alignment handhaben, so daß falsches Alignment nicht eintritt. Charlie Heath, einer der Leiter des ARP-Projekts, ist der Meinung, daß die Ursache für das BPTR-Problem eher von der besonderen BCPL-Implementierung, die Metacomco verwendete, und nicht von den Unzulänglichkeiten der Sprache selbst herrührt.

BSTRs sind grundlegend anders aufgebaut als C-Strings. Das erste Byte eines BSTR gibt die Länge des Strings an, gefolgt vom eigentlichen String ohne Endzeichen. Da zum Definieren der String-Länge nur ein Byte verwendet wird, ist ein BSTR auf 255 Zeichen begrenzt. Ein C-String hingegen kann eine unbegrenzte Zeichenzahl umfassen, wobei das Ende jeweils durch ein Endzeichen markiert wird. Schließlich wird ein BCPL-Stack von der Speicherbasis nach oben angelegt, also genau umgekehrt wie beim 68000-Prozessor, dessen Stack von der Speicherbasis nach unten wächst. Der BCPL- Stack ist also abfallend, während der 68000er-Stack aufsteigend ist.

Aufgrund dieses Sprachenproblems sind Programmierer gezwungen, mit zwei verschiedenen Konventionen umzugehen — einer zum Adressieren der „dos. library" und einer zum Adressieren des restlichen Betriebssystems. Dieser Umstand kann zu Verzögerungen in der Entwicklung und zu einer Erhöhung der Programmierkosten führen. Dieser verzwickte Code bot Grund genug, ARP zu entwickeln, und Heath meinte hierzu: „Die Datenschnittstellen sind alle unterschiedlich, doch das war nur die Hälfte des Problems, nur eine verflixte Hälfte".

Die andere Hälfte des Problems ist Heath zufolge darin zu sehen, daß der Code in der „dos.library" und in der BCPL-Library einfach unzureichend ist. Die BCPL-Library war nicht dokumentiert worden und konnte nur von den BCPL-Befehlen, nicht jedoch von anderen Programmen verwendet werden. Von den Programmierern wird erwartet, daß sie zwei Konventionen parallel beibehalten und ihren Arbeitsaufwand verdoppeln.

Eines der Hauptziele, das sich die ARP-Programmierer gesetzt hatten, war die Behebung der Abhängigkeit von der BCPL-Library. Dies sollte durch das Ersetzen aller BCPL-Befehle und durch das Integrieren einer Library mit Funktionen erfolgen, die von allen Programmen aufgerufen werden können. Es ist ihnen gelungen — und das Ergebnis ist sogar kompakt und elegant.

(Jeffrey Blume)

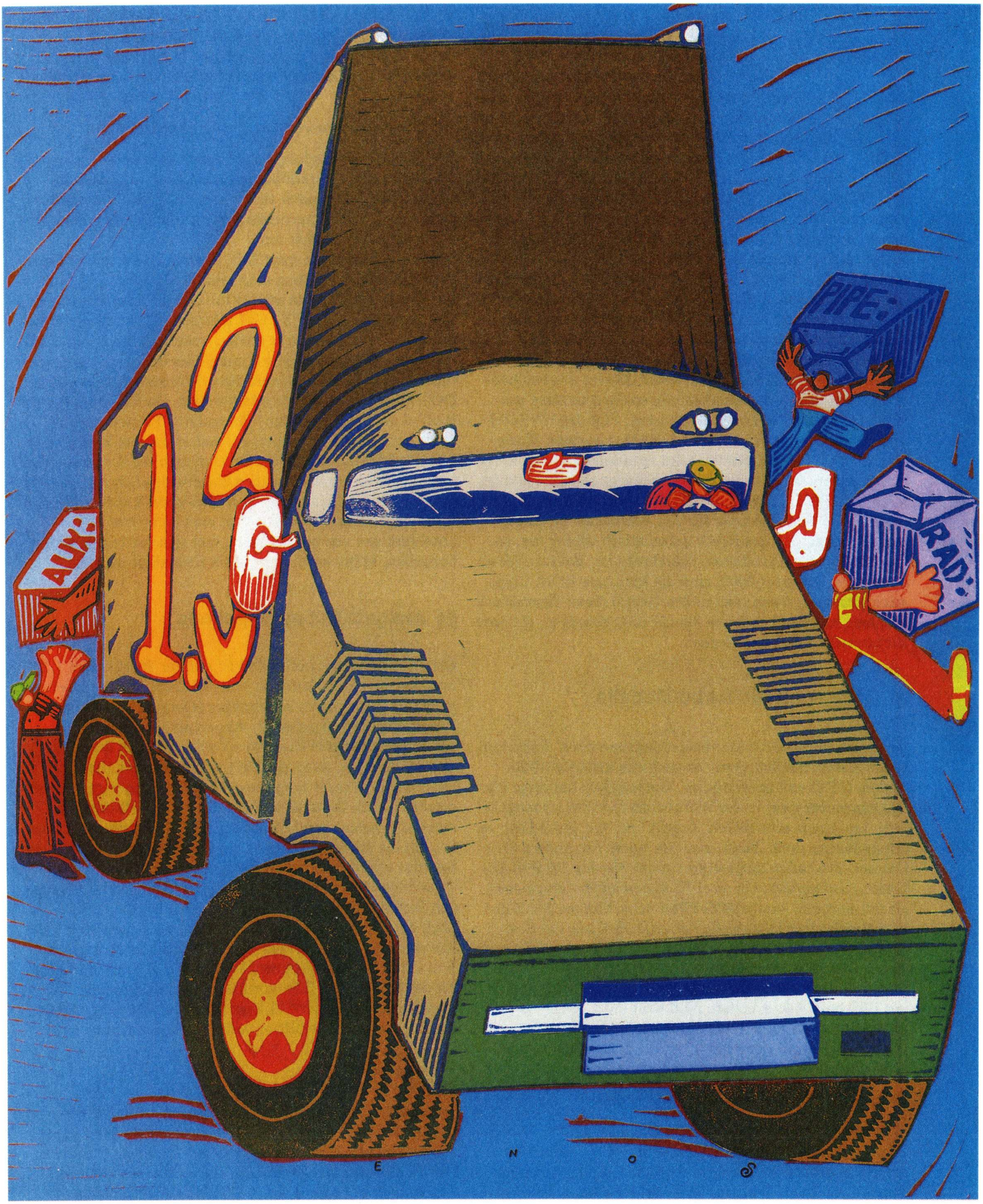
Schlagkräftige Device-Truppe dank Workbench 1.3

Mit der Vielfalt an Gerätetreibern und Systemroutinen der neuen Workbench-Version 1.3, läßt sich der Amiga nun hard- und softwaremäßig noch besser ausstatten. Die nichtphysikalischen Geräte AUX: und PIPE: eröffnen dem User neue Möglichkeiten.

Mit den „alten“ Devices der 1.2-Version des Betriebssystems konnte der MOUNT-Befehl einfachere periphere Operationen zwar problemlos bewältigen, doch schöpfte er die Flexibilität des Amiga beim Installieren neuer Devices nicht aus. Zusammen mit dem Mountlist-Textfile im DEVS:-Verzeichnis kann dieser Befehl nun bis auf einige Ausnahmen alle Hardware- und Software-Devices dem System hinzufügen. Die installierbaren Gerätetreiber veranlassen AmigaDOS, mit externen Hardwaregeräten (zum Beispiel mit Hard-Disk) zu kommunizieren. Die Devices umfassen ebenfalls Software-Treiber, die bestehende Systemquellen auf neue Weise nutzen, ähnlich wie dies beim RAM-Disk-Device der Fall ist. Der Mountlist-File beschreibt verschiedene Device-Attribute und enthält die Information dazu, in welchem File der Gerätetreiber oder die Softwareroutine vorhanden ist.
In der Version 1.2 wurde potentielle Leistung dadurch vergeudet, daß hier nur Mountlist-Einträge für die Installation des A1020 5¼-Zoll-Laufwerkes als AmigaDOS-Gerät sowie ein hypothetisches Gerät mit der Bezeichnung AUX: (das seriellen Geräten ähnlich ist) vorhanden waren.
Mit der Veröffentlichung der Workbench-Version 1.3 macht Commodore die Leistungsstärke des MOUNT-Befehls durch eine breite Palette an mitgelieferten Standard-Devices noch besser nutzbar. Um den Datenzugriff zu beschleunigen, fügte man eine resetfeste RAM-Disk und ein Fast-File-System ein. Die neuen Routinen PIPES: und AUX: erlauben die Kommunikation mit anderen Befehlen und Terminals, während SPEAK: sich mit dem Benutzer unterhält. Für den Fall, daß einem die neuen Befehle nur schwer aus dem Handgelenk fließen, hat Commodore einen Befehlszeilen-Editor integriert.
Besonders interessant ist das neue RAD:-Device, eine zweite RAM-Disk, das wie das altbekannte RAM-Device einen Teil des Computer-Arbeitsspeichers als elektronisches Laufwerk einsetzt. RAD: soll keinen Ersatz für RAM: darstellen, sondern kann höchstens als Ergänzung betrachtet werden. RAM: ist ein Bestandteil von AmigaDOS und wird dadurch angelegt, daß man sich in einem AmigaDOS-Befehl darauf bezieht. Für den Zugriff auf RAD: muß man dieses Device mit dem MOUNT-Befehl in das System integrieren, wobei sicherzustellen ist, daß der File mit dem Gerätetreiber „ramdrive.device“ im DEVS:-Verzeichnis enthalten ist. Das RAM: wird automatisch vergrößert, wenn der Inhalt durch neue Dateien anwächst. Im Gegensatz dazu ist RAD: durch eine bestimmte Größe gekennzeichnet, die beim Installieren mit Mountlist definiert wird.

Mit 1.3 das RAD: neu erfunden

Aufgrund der festgelegten Größe übernimmt RAD: eher die Funktion einer Floppy-Disk als die einer RAM:-Disk. Wie auch die standardmäßigen 3½-Zoll-Disketten ist das RAD: doppelseitig beschreibbar, mit einer Strukur von 512 Bytes pro Sektor und 11 Sektoren pro Spur. Jede Spur (im englischen Cylinder genannt) nimmt 11 KByte Speicherplatz ein (512 Bytes je Sektor × 11 Spuren je Sektor × 2 Seiten). AmigaDOS bestimmt die Anzahl der Spuren, die von RAD: belegt werden, über die LowCyl- und HighCyl-Einträge in der Mountlist. So ordnet zum Beispiel die Angabe von LowCyl = 0 und HighCyl = 21 dem RAD: insgesamt 22 Spuren zu, wobei jeweils 11 KByte je Spur und insgesamt 242 KByte Speicher beansprucht werden. Dies reicht aus, um alle Files des C-Verzeichnisses der Workbench darin unterzubringen. Steht genügend Fast-Memory auf dem Amiga zur Verfügung, könnte man den Wert für HighCyl sogar auf 79 festsetzen und so ein 880 KByte RAD:-Laufwerk erhalten, was dem Speicherumfang einer Diskette entspricht. Besitzt das RAD:-Laufwerk die Speicherkapazität einer Diskette, hat man die Möglichkeit, mit DISKCOPY einen ganzen Disket- ▶



Illustriert von Randall Enos



teninhalt ins RAD: oder umgekehrt zu kopieren. Das RAD:-Laufwerk läßt sich sogar formatieren.

Ein wesentlicher Unterschied zum RAM: liegt in der Beständigkeit des RAD:-Inhalts. Da beide Devices die Sicherung von Daten im Computerspeicher nur simulieren, gehen beim Ausschalten des Gerätes die Inhalte beider Devices verloren. Bei einem Warmstart sieht es jedoch anders aus. Während der gesamte RAM:-Inhalt verloren geht, wenn man die Tastenkombination CTRL-Amiga-Amiga drückt oder eine Guru-Meldung erhält, ist der Inhalt von RAD:, der resetfesten RAM:-Disk, weiterhin geschützt. Sofern der Fehler, der die Guru-Meldung verursacht hat, den Inhalt des Speichers nicht vollends durcheinandergeworfen hat, kann man das RAD: mit Hilfe des ROMs der Kickstart-Version 1.2 erneut installieren und so den Inhalt unversehrt wiederherstellen. Darüber hinaus läßt sich der Amiga mit der Kickstart-Version 1.3 vom RAD:-Device aus neu booten. Neben der Boot-Fähigkeit von solchen Geräten wie Hard-Disks und Netzwerk-Karten aus, fügt der 1.3 ROM-Chip die Eigenschaft hinzu, vom RAD: aus einen Warmstart durchzuführen. Hierbei wird vorausgesetzt, daß RAD: installiert wurde, und daß das Betriebssystem keine bootbare Diskette im Laufwerk df0: findet. Selbst wenn das System eine Boot-Diskette für den Warmstart benötigt, stellt Kickstart 1.3 das RAD: automatisch wieder her, so daß ein erneutes Installieren überflüssig ist. Besitzen Sie einen Amiga 500 oder 2000, sind die Kickstart-ROM-Chips auszuwechseln, wenn man in den Genuß dieser Eigenschaft kommen möchte; bei einem Amiga 1000 genügt es, die neue Kickstart-Diskette einzulegen.

RAD: auch für Turbo-Modelle

Da es sich beim RAD: um einen einsetzbaren und formatierbaren Zusatz handelt, kann es auch das neue Fast-File-System (FFS) voll unterstützen. Commodore hat diese File-Systemroutine zwar in der Workbench-Version 1.3 mitgeliefert, um einen schnelleren Zugriff auf die Hard-Disk zu ermöglichen, doch kann man sie auch für andere Disk-Devices verwenden, abgesehen einmal von den 3½-Zoll-Disketten-Laufwerken. Da das FFS Daten anders abspeichert als das normale AmigaDOS-Filesystem, können 3½-Zoll-Diskettenlaufwerke aktuelle AmigaDOS-Disks mit dem neuen System nicht lesen. Bei Zusatzgeräten, die nicht auswechselbar sind, wie Hard-Disks oder RAM-Disks, wird man mit diesem Problem nicht konfrontiert. Möchte man mit FFS im RAD: arbeiten, muß man unter RAD: im DEVS:MOUNTLIST-File zwei Dinge hinzufügen.
Hinter RAD: und vor dem #, mit dem die Eingabe abgeschlossen wird, ist jeweils folgendes zu tippen:

```
GlobVec=-1FileSystem=1:FastFileSystem
```

Weiterhin muß gewährleistet sein, daß der Fast-File-System-File, der in der zweiten Zeile erscheint, im 1:-Verzeichnis auf der Workbench enthalten ist. Nachdem die Mountlist-Einträge verändert wurden, installiert man das Laufwerk mit dem Befehl MOUNT RAD. Da man hier nicht auf das Standard-File-System zurückgreift, muß das RAD: vor dem Arbeiten formatiert werden. Dieser Arbeitsgang läßt sich durch die QUICK-Option des FORMAT-Befehls erheblich verkürzen:

```
:SYSTEM/FORMAT drive RAD: name SpeedyQUICK
```

Obwohl die Verwendung des FFS auf der auswechselbaren RAM-Disk die Operationsgeschwindigkeit in gewissem Maße erhöht, es handelt sich ja um ein RAM-Laufwerk —, weist RAD: auch ohne dieses System eine ziemlich hohe Reaktionsgeschwindigkeit auf.
Das Fast-File-System spart Zeit, doch es kostet seinen Preis. Unter Kickstart 1.3 kann man vom RAM-Laufwerk aus nicht neu booten, wenn man es mit dem FFS formatiert hat. Bei Verwendung des 1.3 ROM läßt sich nicht einmal der Inhalt des RAM-Laufwerks retten, wenn man nach dem Formatieren mit dem FFS neu bootet. Dies ist darauf zurückzuführen, daß dieses Laufwerk beim Warmstart als normales DOS-File-System-Device geladen wird. Da Kickstart 1.3 ein RAM-Laufwerk im „alten“ AmigaDOS-Format voraussetzt, wird davon ausgegangen, daß es sich bei RAD: nicht um eine DOS-Disk handelt. Benutzt man die Kickstart-Version 1.2, hat man jedoch einen eingebauten Schutz, weil das Betriebssystem ein erneutes Installieren des Laufwerks beim Warmstart verlangt. Auch wenn man nicht vom RAM-Laufwerk aus neu booten kann, läßt sich der Inhalt wiederherstellen, selbst wenn es mit FFS formatiert wurde.

AmigaDOS baut Pipelines

Die Device-Systemroutine PIPE: auf der Workbench 1.3 emuliert die Pipes-Funktion von MS-DOS, die die Übertragung eines Programm-Ergebnisses (Output) in ein anderes Programm (Input) gestattet. Angenommen, man möchte ein umfangreicheres Disketten-Verzeichnis auf dem Bildschirm anzeigen, ohne mit DIR zu arbeiten, da dieses die File-Namen in einer Folge ausgibt und die Anzeige bei vollgeschriebenem Bildschirm nicht unterbricht. Indem man nun das Ergebnis von DIR an das MORE-Programm umleitet, das die Anzeige nur eines einzigen Bildschirms mit Text bewirkt, erhält man die abgefragte Information im gewünschten Format. Unter UNIX oder MS-DOS würde man den Befehl DIRNMORE eingeben. Da aber die Amiga-Shell den Operanden N nicht erkennt, muß man eine Pipeline simulieren, um zu dem gleichen Ergebnis zu gelangen. Wie schon bei der Workbench-Version 1.2 kann man auch bei der 1.3-Version Files vorübergehend in einen anderen Speicherbereich der RAM-Disk umleiten. Zur Übertragung des Ergebnisses von DIR an MORE gibt man ein:

```
DIR >ram:temp
MORE ram:temp
DELETE ram:temp
```

Eine noch wirksamere Methode, besonders bei umfangreicheren Files, besteht darin, PIPE: zu installieren, dessen Routine sich im File „1:Pipe-Handler“ befindet. Der PIPE:-Device übernimmt die Funktion einer „Pipeline“, mit der das Ergebnis eines Programms (OUTPUT) in ein anderes

Programm (INPUT) umgeleitet wird. In einem Programmabschnitt werden Informationen an die Pipeline geschrieben, bevor ein willkürlicher Name für sie vergeben wird (zum Beispiel pipe:temp). Jeder Pipeline-Name verbraucht 4 KByte Buffer, was eigentlich nur aussagt, daß äußerst viele Informationen an die Pipeline geschrieben werden können, bevor der Vorgang blockiert. Wenn der zweite Programmabschnitt den 4-KByte-Buffer liest (indem er auf denselben File zugreift, in den die Information geschrieben wurde), kann das erste Programm weitere 4 KByte Daten schreiben, solange, bis das gesamte Ergebnis übertragen worden ist. Nachdem die Anweisung MOUNT PIPE erteilt wurde, kann der Output von DIR mit der nachstehenden Befehlsfolge an MORE „gepumpt" werden:

```
RUN DIR >pipe:temp
MORE pipe:temp
```

Es ist zu beachten, daß man mit RUN einen separaten DIR-Vorgang auslösen kann. Beide Befehle können nicht denselben CLI-Vorgang verwenden, denn wenn der Output des Verzeichnisses größer ist als 4 KByte, wird das Programm von DIR nicht beendet und die Eingabeaufforderung des CLI nicht angezeigt, bevor MORE den gesamten Output gelesen hat.

Der Umweg, den PIPE: zum Simulieren von Pipelines benutzt, mag vielleicht länger und umständlicher sein als bei anderen Systemen, doch sind einige wesentliche Vorteile nicht zu verkennen. So kann man sich neben der oben beschriebenen herkömmlichen Transfer-Funktion auch ausschließlich die Buffer-Fähigkeiten von PIPE: zunutze machen. Viele Terminalprogramme laden Files zum Beispiel synchron ein, indem sie einen Datenblock empfangen und diesen an die Disk senden, dann abwarten, bis der Schreibvorgang beendet ist, und anschließend nach dem nächsten Datenblock fragen. Jeder Zwischenschritt beim Schreiben verzögert die Übertragung. Diese Verzögerung kann man durch Downloading an einen File im RAM: vermeiden, doch geht man hier das Risiko ein, daß die RAM-Disk vollgeschrieben ist, noch bevor der File-Transfer beendet ist, oder daß man vergißt, den File auf Diskette zu kopieren, bevor man den Computer ausschaltet. Hier bietet sich als Lösung der Befehl

```
COPY pipe:temp TO df0:downfile
```

an, den man vor der Ausführung des Terminalprogrammes eingibt. Anschließend wird an den File pipe:temp geladen. Dank dieser Vorgehensweise ist es möglich, größere Datenmengen zwischenzuspeichern, bevor sie geschrieben werden, und so die Verzögerungen zu reduzieren. Gleichzeitig werden die Risiken, die mit Downloading ins RAM: verbunden sind, ausgeschaltet. Wenn der Vorgang abgeschlossen ist, befindet sich der File bereits sicher auf Diskette.

Neues AUX: mit mehr Power

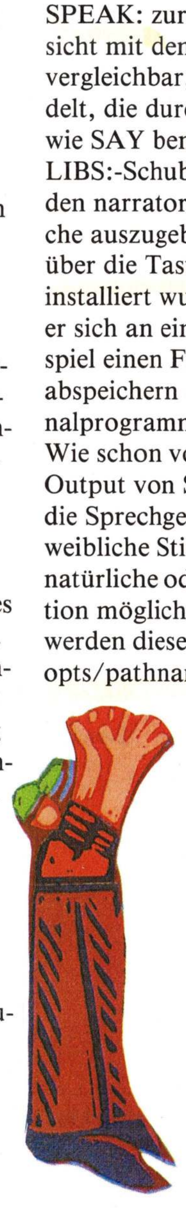
Will man Daten an ein anderes Terminal übermitteln, kommt einem das neue Device AUX: entgegen, dessen Routine sich im File „1:Aux- Handler" befindet. AUX: überträgt Daten über den Serial Port, ähnlich dem SER:-Device; der Unterschied liegt hier beim Buffering. Während SER: seinen Output zwischenspeichert und ihn erst dann sendet, wenn sich ein 512-Byte-Block angesammelt hat, überträgt AUX: ohne Buffering über den Serial Port. Bei AUX: ohne Buffering kann man ein CLI-Fenster erstellen, das den Serial Port für Input und Output benutzt. Man installiert den AUX:-Treiber und gibt ein:

```
NEWCLI AUX:
```

Auf diese Weise läßt sich ein weiterer Computer oder ein weiteres Terminal an den Amiga anschließen, so daß AmigaDOS-Befehle von der zweiten Maschine aus über den Serial Port oder sogar über ein Modem eingegeben werden können. Man kann natürlich kein Programm auf Window-Intuition-Basis ausführen, doch ist es möglich, CLI-Befehle wie zum Beispiel DIR und INFO auf dem Fern-Terminal zu verwenden, mit dem Zweck, Informationen über die Amiga-Disks zu erhalten. Mit Hilfe des TYPE-Befehls lassen sich Files an einen entfernten Bildschirm senden, wo sie von einem Buffer-File entgegengenommen werden. Obschon CLIs mit Serial Port den Amiga nicht völlig zu einem Mehrplatzsystem umzuwandeln vermögen, ist man davon dennoch nicht sehr weit entfernt.

Um die eingebaute Sprachsynthese des Amiga besser zugänglich zu machen, fügte Commodore die Routine SPEAK: zur Workbench 1.3 hinzu. SPEAK ist in der Hinsicht mit dem SAY-Programm in der Utilities-Schublade vergleichbar, daß es eingegebenen Text in Sprache umwandelt, die durch die Audio-Kanäle ausgegeben wird. Genau wie SAY benutzt es den translator.library-File aus der LIBS:-Schublade, um Text in Phoneme umzusetzen, sowie den narrator.device aus DEVS:, um die phonetische Sprache auszugeben. Während SAY ausschließlich die Eingabe über die Tastatur erlaubt, verarbeitet SPEAK, nachdem es installiert wurde, den Input aus beliebigen Quellen, sofern er sich an einen File schreiben läßt. So kann man zum Beispiel einen File aus einer Textverarbeitung nach SPEAK: abspeichern oder SPEAK als Empfangs-File für ein Terminalprogramm öffnen.

Wie schon von SAY her bekannt, läßt sich auch der Sound-Output von SPEAK regeln. Man kann den Pitch-Level und die Sprechgeschwindigkeit festlegen sowie männliche oder weibliche Stimm-Merkmale wählen. Darüber hinaus ist eine natürliche oder roboterähnliche, monotone Stimm-Modulation möglich. Zum Hinzufügen von Stimm-Eigenschaften werden diese beim Zugriff auf diesen Device in die SPEAK: opts/pathname-Anweisung integriert. Als Beispiel wollen wir uns einen File anhören, der von einer weiblichen Stimme mit einem Pitch-Level von 200 wiedergegeben wird:



```
COPY filename to SPEAK:opts/f/p200
```

Im Nachhinein die vollständige Liste der Stimmoptionen, die sich in den SPEAK:opt s/ pathname integrieren lassen:

```
P### Set Pitch (### steht für eine Zahl
zwischen 65 und 320)

S### Set Speed (### steht für eine Zahl
zwischen 30 und 400).
```

- M Männliche Stimm-Merkmale
- F Weibliche Stimm-Merkmale
- R Robotersprache (monoton und ohne Stimm-Modulation)
- N Natürliche Sprache (natürliche Stimm-Modulation)
- O0 Optionen zur Regelung des Input nicht möglich
- O1 Optionen zur Regelung des Input möglich
- A0 Phonem-Eingabemodus ausschalten
- A1 Phonem-Eingabemodus einschalten
- D0 Satzpausen nur durch Punktsetzung kennzeichnen
- D1 Satzpausen durch Umschalten (Carriage Return) und Zeilenvorschub sowie durch Punktsetzung kennzeichnen.

Super-Con-Man garantiert flotte Edition

Obschon der Amiga zum Benutzer spricht, ist er nicht imstande, die Behebung eines fehlerhaft formulierten Befehls zu erklären. Eine häufig vorgebrachte Kritik in bezug auf die CLI-Umgebung gilt der mangelnden Unterstützung von Befehlszeilen-Edition im Konsolfenster. Schleicht sich im ersten Wort einer Befehlszeile ein Tippfehler ein, mußte bisher die gesamte Zeile gelöscht und neu geschrieben werden.



Der neue 1.3-Device NEWCON: (vergleichbar mit dem Shareware-Programm ConMan) stellt ein Konsolfenster zur Verfügung, das nicht nur das Editieren mit den Cursortasten erlaubt, sondern auch noch einen Befehls-History-Buffer von 2 KByte hinzufügt. Nachdem man das NEWCON: -Device installiert hat (die Routine befindet sich im File l:Newcon-Handler), läßt sich ein CLI-Fenster öffnen, das diesen neuen Konsol-Device benutzt. Hierzu gibt man ein:

```
NEWCLI NEWCON:x/y/w/h/name
```

Die Buchstaben x und y definieren die Position der linken oberen Ecke des Fensters; w und h geben die Höhe und Breite in Pixel wieder, und name steht für einen frei zu wählenden Fensternamen.

Mit einem NEWCON:-Fenster ist es möglich, eine Befehlszeile mit den Pfeil-links- und Pfeil-rechts-Tasten zu editieren, wenn der Cursor jeweils um ein Zeichen versetzt werden soll oder aber mit den Shift-Tasten, die den Cursor an den Anfang beziehungsweise das Ende der Befehlszeile positionieren. Die Pfeile oben und unten implementieren eine Befehls-History-Funktion.

Jedesmal, wenn eine Befehlszeile eingegeben wird, wird die Zeile in einem Circle-Buffer von 2 KByte gespeichert. Das Drücken von Pfeil-oben holt die letzte Eingabe in den Buffer zurück, die daraufhin in der Eingabezeile erscheint. Durch das Drücken von Pfeil-unten kann man sich vorwärts durch den Buffer bewegen. Die Tastenkombination Shift/Pfeil-unten bringt einen an das Ende des Buffers. Möchte man nicht jeden vorhergehenden Befehl durchgehen, kann man auf die Suchfunktion der Befehls-History zurückgreifen. Das Tippen eines Teils der Befehlszeile mit anschließender Betätigung von Shift/Pfeil-oben, löst die Suche nach der letzten Befehlszeile aus, die dieser Zeichenfolge entspricht.

Während Commodore zunächst nur beabsichtigte, mit dem 1.3 Betriebssystem eine erweiterte Drucker-Unterstützung zusammen mit der Selbstboot-Funktion und einem schnelleren Zugriff für Festplatten zu liefern, wurden noch ein paar Bonbons mehr eingebaut. Neue Devices wie das resetfeste RAM-Laufwerk, der Speech-Device, die Pipes-Routine erhöhen die Verarbeitungsgeschwindigkeit und vergrößern die Reichweite des Systems, ohne daß man dafür mehr als ein paar Befehle investieren muß.

(Sheldon Leemon/Sabine Bröker/uk)

Abtippen jetzt noch leichter mit checksum!

Nun liegt checksum auch in AmigaBASIC vor! Mit den verbesserten Versionen für Assembler, C und BASIC werden Tippfehler bei AmigaWelt-Listings zur Vergangenheit gehören.

Viel Arbeit bei der Fehlerkorrektur nach dem Eintippen von Listings aus AmigaWelt erspart man sich mit „checksum". Hierzu sollte ein Listing — mit Ausnahme von sogenannten Hexdumps, dafür ist NanoMon zuständig — einfach mit einem beliebigen ASCII-Editor eingegeben werden. Das kann der „ED" der Workbench-, der „MicroEMACS" der Extras-Diskette oder einer der unzähligen sonstigen kommerziellen oder frei erhältlichen Editoren sein. Auch im Falle von BASIC-Programmen, die ja üblicherweise direkt im Interpreter eingegeben werden, sollte die Eingabe auf diese Weise erfolgen. Sie werden feststellen, daß dies gegenüber dem sehr langsamen AmigaBASIC-Editor die Arbeit durchaus erleichtern kann. Sollten Sie dabei aus Gewohnheit die Groß- und Kleinschreibung nicht beachtet haben (dafür sorgt ja sonst AmigaBASIC), so laden Sie das Programm einfach durch ...

```
LOAD "Programmname"
```

... von AmigaBASIC aus ein und speichern es direkt anschließend mit ...

```
SAVE "Programmname",A
```

... wieder ab. Die Option "A" ist dabei sehr wichtig, da checksum nur mit ASCII-Dateien arbeitet und die komprimierte Ablage in Tokens, die AmigaBASIC ansonsten vornimmt, nicht versteht.
Bei der Eingabe können die sprachspezifischen Kommentare inklusive der Begrenzer, also etwa „REM" und das Hochkomma (Alt/Ä) in BASIC oder "/* ... */" in C, völlig weggelassen werden. Dies ist ein wichtiger Unterschied zur früheren Version, die auf einen (nicht erlaubten) sehr dubiosen Nebeneffekt mancher C-Compiler Rücksicht genommen hat. Da dieses Verschmelzen von Identifiers jedoch mit den C-Compilern für den Amiga ohnehin nicht funktioniert und mittlerweile auch durch den neuen ANSI-Standard verboten wurde, ist das Verhalten von checksum entsprechend geändert worden, um die Eingabe weiter zu erleichtern.
Durch diesen Unterschied müssen Sie für alle Programme ab AmigaWelt 4/88 (soweit nicht anders vermerkt) unbedingt die neue Version von checksum verwenden!
Ebenfalls ohne Bedeutung für die Prüfsumme sind Leerzeilen, Einrückungen und mehr als ein Leerzeichen auf einmal, da diese von checksum völlig ignoriert werden. Die beiden letzteren Fälle werden dabei aber noch darauf untersucht, ob sie innerhalb von Texten vorkommen, denn schließlich kann sich in C ein String über mehrere Zeilen erstrecken. Hier ist die Anzahl an Leerzeichen dann nämlich durchaus wichtig. Man kann also nicht einfach führende Leerstellen ignorieren! Zum Glück ist checksum intelligent genug, diese und auch andere vertrackte Fälle zu erkennen und entsprechend zu reagieren.
Zwei oder mehr Leerzeichen innerhalb von Textkonstanten, meist durch einfache oder doppelte Anführungszeichen begrenzt, werden durch ...

```
[xSPACES]
```

... dargestellt. „x" steht dabei natürlich für die Anzahl. Sie ersparen sich dadurch mühsames Zählen nicht sichtbarer Zeichen.
Die Prüfsumme selbst befindet sich am Ende einer Zeile und besteht aus drei Ziffern. Dies bietet hinreichend Redundanz, nicht zuletzt auch deshalb, da checksum aufgrund der Unterscheidung der drei unterstützten Programmiersprachen selbst erkennt, welche Zeichen für ein Programm bedeutend sind und welche nicht. Bei der Eingabe muß kein Abstand zwischen der eigentlichen Zeile und den drei Prüfziffern eingehalten werden, sie können direkt im Anschluß an den eigentlichen Quelltext eingegeben werden — natürlich stören Spaces auch nicht.
Das einzige, was Sie bei allen Sprachen nicht miteingeben dürfen, sind die Zeilennummern zu Beginn einer jeden Zeile. Diese dienen lediglich der Übersicht und dazu, fehlerhafte Stellen zu bezeichnen.
Ebenfalls Aufpassen heißt es bei Zeilen ohne Zeilennummer, denn diese gehören zur vorherigen Zeile und mußten aus Layout-technischen Gründen getrennt werden. Dabei wird vorzugsweise an Leerstellen getrennt, sofern möglich. Ob und wieviele Leerzeichen nun an das Ende der vorherigen Zeile gehören, erkennen Sie daran, um wieviele Stellen (meist eine oder gar keine) die Fortsetzungszeile gegenüber dem ersten Teil (mit Zeilennummer) eingerückt wurde.
Sollten Sie das Listing vollständig eingetippt haben, bis hierhin können Sie alles mit Ihrem Lieblings-ASCII-Editor erledigen, so wird diese Datei an checksum übergeben.
Doch wie wird checksum selbst zum Laufen gebracht? Hier muß zwischen der Version in C und der in BASIC unterschieden werden. Die BASIC-Version wird einfach in den Interpreter geladen und gestartet.

Nur unwesentlich schwieriger wird es bei der C-Version, die erfolgreich mit Lattice-C-4.01 und Manx-Aztec-C-3.6a compiliert worden ist. Hier müssen Sie zumindest die drei Dateien ...

```
CheckLib.h
CheckLib.c
CheckSum.c
```

... vorliegen haben. Als Besitzer von Lattice-C-4.00 und der neuen Version des MetaComCo-Toolkit erleichtern Sie sich die Arbeit, wenn außerdem ...

```
Rules.make
MakeFile
```

... im Arbeitsverzeichnis vorliegen. Dann geben Sie nämlich nur noch ...

```
Make
```

... ein, und nach kurzer Zeit, sofern kein Fehler aufgetreten ist, liegt das Programm „checksum" fertig vor! Sollte Ihnen die Benutzung von MakeFiles nicht geläufig sein, so geben Sie als Benutzer von Lattice-C ein ...

```
LC1 -ceft -w CheckLib
LC2 -v CheckLib
LC1 -ceft -w CheckSum
LC2 -v CheckSum
BLink LIB:c.o+CheckSum.o+CheckSum.o TO CheckSum LIB
 LIB:lcs.lib+LIB:amiga.lib SC SD ND
```

Falls Sie noch nicht über die Version 4 von Lattice verfügen, so lassen Sie die Option „-w" in den mit „LC1" beginnenden Zeilen einfach weg und verwenden beim Linken anstelle von „lcs.lib" einfach „lc.lib".
Für Aztec-Benutzer lauten die Befehle einfach ...

```
cc CheckLib
cc CheckSum
ln CheckSum.o CheckLib.o -lc
```

Am schnellsten ist natürlich in jedem Falle die C-Version von checksum. Trotzdem läßt sich auch die BASIC-Version mit dem AC-BASIC-Compiler erheblich beschleunigen.
Die abgespeicherte ASCII-Datei wird nun mit dem Befehl „checksum" auf ihre Richtigkeit überprüft. „checksum" sollte sich dabei im C-Directory befinden. Die Syntax lautet:

```
CheçkSum [-l<sprache>] [-o<ausgabe>]
         [-t<start>] [-s<schritt>] <quellfile>
```

<sprache> Je nach Typ des Programms entweder „C", „BASIC" oder „ASSEM", Default ist „C". Batchdateien und ähnliche werden als Assembler abgedruckt.

<ausgabe> Name des Ausgabefiles, Default ist die Standardausgabe (CLI-Fenster). Rein theoretisch kann hier jedes Amiga-spezifische Gerät angegeben werden, meist wird dies ein Laufwerk mit Pfad und Filename sein.

<start> Tabulatorstart (erste Position, 1 bis 64), Default ist 1.

<schritt> Tabulatorschrittweite (Abstand, 1 bis 32), Default ist 8.

<quellfile> Das zu überprüfende Quellfile.

Die Reihenfolge der Parameter spielt keine Rolle. Nach dem Start der BASIC-Version werden Sie nach den gleichen Daten gefragt, Sprach-Default ist hierbei jedoch sinnvollerweise BASIC.
Wurde ein Assembler-Quelltext etwa unter dem Namen „quell.sum" gespeichert, so wird durch ...

```
CheckSum -lassem -oquell.asm quell.sum
```

... das eigentliche Quellfile erzeugt. Dabei werden gleichzeitig die Prüfsummen entfernt und jede Zeile kontrolliert. checksum erkennt Fehler in der Programmzeile, auch Auslassungen und Dreher. Selbst fehlende Programmzeilen werden gemeldet. Die Fehlermeldungen im einzelnen:
„no checksum": Die Zeile enthält keine Prüfsumme. Entweder wurde sie vergessen oder eine getrennt gedruckte Zeile irrtümlich getrennt eingegeben. Obwohl die Prüfsumme von vorhergehenden Zeilen abhängt, schleppt sich dieser Fehler NICHT bis zum Ende mit.
„checksum error": Entweder eine falsche Prüfsumme oder die vorhergehende Zeile fehlt.
„unexpected end of file": Die Verarbeitung traf innerhalb von Kommentaren, Anführungszeichen oder ähnlichem auf das Dateiende.
„too many tabs": Die Zeile enthält so viele Tabulatoren, daß der interne Puffer überlief. Bei einer Puffergröße von 512 Bytes kann es nur unter sehr mysteriösen Umständen zu diesem Fehler kommen.
Beim Auftreten solcher Fehler geht es wieder zurück in den Editor. Korrigieren Sie die gemeldeten Zeilen, und lassen Sie den Quelltext erneut überprüfen.
Wer unsicher beim Abtippen oder einfach zu faul ist, sollte sich die AmigaWelt-Leserdiskette besorgen, auf der sich die fertigen Versionen von checksum befinden.
(Ralph Babel)

Programmname: checksum
Sprache: BASIC

Auf der Leserdiskette befindet sich nur das lauffähige C-Compilat!

ABC **Eingabe mit beliebigem Editor (ohne Zeilennummern) Überprüfen mit checksum (alte Version)**

```
 1 REM                                                     496
 2 REM "CheckSum.bas - Update 08-Apr-1988 für AmigaBASIC   672
 3 REM "Copyright (C) 1988 CW-Publikationen, München       272
 4 REM "all rights reserved - alle Rechte vorbehalten      064
 5 REM                                                     840
 6                                                         840
 7 DEFINT a-z                                              022
 8                                                         022
 9 ' "Bereich für Prüfsumme (0 .. SUMRANGE-1)              325
10                                                         325
11 SUMWIDTH = 3                                            214
12 SUMRANGE = 1000                                         542
13                                                         542
14 ' "unterstützte Sprachen (lmode)                        941
15                                                         941
16 L.C     = 1                                             396
17 L.BASIC = 2                                             551
18 L.ASSEM = 3                                             622
19                                                         622
20 ' "Bitpositionen (qmode)                                981
21                                                         981
22 Q.ENCLOSED = &O1                                        756
23                                                         756
```

```
 24 Q.IGNORE = &O2                                                    191
 25                                                                   191
 26 Q.DISCARD  = &O14                                                 083
 27 Q.DISCARD1 = &O4                                                  267
 28 Q.DISCARD2 = &O10                                                 821
 29 Q.DISCARD3 = &O14                                                 808
 30                                                                   808
 31 Q.C.MAINMODES = &O60                                              177
 32 Q.C.SINGLE    = &O20                                              027
 33 Q.C.DOUBLE    = &O40                                              788
 34 Q.C.COMMENT   = &O60                                              847
 35 Q.C.NONWHITE  = &O100                                             794
 36                                                                   794
 37 Q.BASIC.DOUBLE    = &O20                                          988
 38 Q.BASIC.DATA[5SPACES]= &O40                                       151
 39 Q.BASIC.REM       = &O100                                         322
 40 Q.BASIC.NONWHITE = &O200                                          769
 41 Q.BASIC.PRECSYM  = &O400                                          657
 42                                                                   657
 43 Q.ASSEM.MAINMODES = &O60                                          040
 44 Q.ASSEM.SINGLE    = &O20                                          092
 45 Q.ASSEM.DOUBLE    = &O40                                          623
 46 Q.ASSEM.COMMENT   = &O60                                          888
 47 Q.ASSEM.NONWHITE  = &O100                                         297
 48 Q.ASSEM.NOTFIRST  = &O200                                         176
 49                                                                   176
 50 ' "Defaults etc.                                                  279
 51                                                                   279
 52 MINSTART      = 1                                                 731
 53 MAXSTART      = 64                                                834
 54 DEFAULT.START = 1                                                 889
 55                                                                   889
 56 MINSTEP      = 1                                                  135
 57 MAXSTEP      = 32                                                 279
 58 DEFAULT.STEP = 8                                                  847
 59                                                                   847
 60 ' "Parameter auswerten                                            810
 61                                                                   810
 62 loop1:                                                            439
 63 INPUT"Sprache (Default = BASIC)";a$:a$=UCASE$(a$)                 449
 64 IF a$="C" THEN                                                    448
 65   lmode=L.C                                                       459
 66 ELSEIF a$="" OR a$="BASIC" THEN                                   113
 67   lmode=L.BASIC                                                   599
 68 ELSEIF a$="ASSEM" THEN                                            925
 69   lmode=L.ASSEM                                                   914
 70 ELSE                                                              643
 71   GOTO loop1                                                      487
 72 END IF                                                            989
 73                                                                   989
 74 loop2:                                                            311
 75 PRINT"Tabulator-Start (Default ="DEFAULT.START")";                160
 76 INPUT tabstart                                                    841
 77 IF tabstart=0 THEN                                                133
 78   tabstart=DEFAULT.START                                          820
 79 ELSEIF tabstart<MINSTART OR tabstart>MAXSTART THEN                642
 80   GOTO loop2                                                      007
 81 END IF                                                            253
 82                                                                   253
 83 loop3:                                                            332
 84 PRINT"Tabulator-Schrittweite (Default ="DEFAULT.STEP")";          156
 85 INPUT tabstep                                                     063
 86 IF tabstep=0 THEN                                                 701
 87   tabstep=DEFAULT.STEP                                            552
 88 ELSEIF tabstep<MINSTEP OR tabstep>MAXSTEP THEN                    558
 89   GOTO loop3                                                      532
 90 END IF                                                            610
 91                                                                   610
 92 INPUT"Quelldatei";srcname$                                        092
 93                                                                   092
 94 INPUT"Zieldatei (Default = Bildschirm)";dstname$                  382
 95 IF dstname$="" THEN dstname$="SCRN:                               400
 96                                                                   400
 97 ' "Files öffnen, Summen prüfen und Quelltext erzeugen             143
 98                                                                   143
 99 OPEN srcname$ FOR INPUT AS 1                                      182
100 OPEN dstname$ FOR OUTPUT AS 2                                     914
101                                                                   914
102 qmode    = 0                                                      969
103 checksum = 0                                                      617
104 nline    = 0                                                      408
105                                                                   408
106 WHILE NOT EOF(1)                                                  586
107   nline=nline+1                                                   943
108   LINE INPUT#1,buffer$                                            889
109                                                                   889
110   ' "Tabulatoren durch Leerzeichen ersetzen                       356
111                                                                   356
112   col=1                                                           408
113   FOR i=1 TO LEN(buffer$)                                         871
114     IF MID$(buffer$,col,1)=CHR$(9) THEN                           785
115       IF col<tabstart THEN                                        806
116         ins=tabstart-col                                          149
117       ELSE                                                        590
118         ins=tabstep-(col-tabstart) MOD tabstep-1                  047
119       END IF                                                      509
120       buffer$=LEFT$(buffer$,col-1)+STRING$(ins+1,"
           ")+MID$(buffer$,col+1)                                     573
121       col=col+ins                                                 711
122     END IF                                                        469
123     col=col+1                                                     270
124   NEXT                                                            685
125                                                                   685
126   GOSUB rmtrlspc                                                  636
127                                                                   636
128   IF LEN(buffer$)<SUMWIDTH GOTO nosum                             845
129                                                                   845
130   givensum=0                                                      916
131   i=3                                                             473
132   loop:                                                           741
133     digit=ASC(LEFT$(RIGHT$(buffer$,i),1))-48                      463
134     IF digit<0 OR digit>9 GOTO nosum                              407
135     givensum=givensum*10+digit                                    515
136   i=i-1:IF i>0 GOTO loop ELSE skip                                555
137                                                                   555
138   nosum:                                                          239
139   givensum=-1                                                     816
140   PRINT "line "nline": no checksum                                747
141                                                                   747
142   skip:                                                           284
143   IF givensum<>-1 THEN                                            828
144     buffer$=LEFT$(buffer$,LEN(buffer$)-SUMWIDTH)                  921
145     GOSUB rmtrlspc                                                624
146   END IF                                                          246
147                                                                   246
148   buffer$=buffer$+CHR$(10)                                        457
149   PRINT#2,buffer$;                                                960
150                                                                   960
151   "irrelevante Zeichen entfernen und Prüfsumme bilden             231
152                                                                   231
153   FOR i=1 TO LEN(buffer$)                                         314
154     c$=MID$(buffer$,i,1)                                          781
155                                                                   781
156     IF qmode AND Q.DISCARD THEN ' "Ausgleich 'look-ahead'
         o. ä.                                                        408
157       qmode=qmode AND NOT Q.DISCARD OR (qmode AND
           Q.DISCARD)-Q.DISCARD1                                      845
158     ELSE                                                          110
159       IF lmode=L.C THEN                                           191
160         IF (qmode AND Q.C.MAINMODES)=Q.C.SINGLE THEN              491
161           IF c$="'" THEN                                          664
162             qmode=qmode AND NOT (Q.ENCLOSED OR
                 Q.C.MAINMODES)                                       340
163           ELSE                                                    869
164             qmode=qmode OR Q.ENCLOSED                             279
165             IF c$="\" THEN                                        841
166               qmode=qmode OR Q.DISCARD1                           729
167             END IF                                                727
168           END IF                                                  333
169         ELSEIF (qmode AND Q.C.MAINMODES)=Q.C.DOUBLE THEN          439
170           IF c$=CHR$(34) THEN                                     938
171             qmode=qmode AND NOT (Q.ENCLOSED OR
                 Q.C.MAINMODES)                                       678
172           ELSE                                                    215
173             qmode=qmode OR Q.ENCLOSED                             929
174             IF c$="\" THEN                                        043
175               qmode=qmode OR Q.DISCARD1                           019
176             END IF                                                425
177           END IF                                                  783
178         ELSEIF (qmode AND Q.C.MAINMODES)=Q.C.COMMENT THEN         849
179           IF MID$(buffer$,i,2)="*/" THEN                          825
180             qmode=qmode AND NOT Q.C.MAINMODES OR
                 Q.DISCARD1 OR Q.IGNORE                               990
181           ELSEIF c$=CHR$(10) THEN                                 180
182             qmode=qmode AND NOT Q.IGNORE                          760
183           ELSE                                                    353
184             qmode=qmode OR Q.IGNORE                               526
185           END IF                                                  524
186         ELSE                                                      925
187           IF c$=" " THEN                                          387
188             IF qmode AND Q.C.NONWHITE THEN                        178
189               qmode=qmode AND NOT Q.C.NONWHITE                    015
190             ELSE                                                  080
191               qmode=qmode OR Q.IGNORE                             561
192             END IF                                                367
193           ELSE                                                    968
194             qmode=qmode AND NOT Q.IGNORE OR Q.C.NONWHITE          705
195                                                                   705
196             IF c$=CHR$(10) THEN                                   334
197               qmode=qmode AND NOT (Q.IGNORE OR
                   Q.C.NONWHITE)                                      240
198             ELSEIF c$="'" THEN                                    510
199               qmode=qmode OR Q.C.SINGLE                           434
200             ELSEIF c$=CHR$(34) THEN                               030
201               qmode=qmode OR Q.C.DOUBLE                           011
202             ELSEIF MID$(buffer$,i,2)="/*" THEN                    752
203               qmode=qmode OR Q.C.COMMENT OR Q.DISCARD1 OR
                   Q.IGNORE                                           284
204             END IF                                                834
205           END IF                                                  160
206         END IF                                                    422
207       ELSEIF lmode=L.BASIC THEN                                   751
208         IF c$=CHR$(10) THEN                                       428
209           qmode=0                                                 575
210         ELSEIF (qmode AND Q.BASIC.REM)=0 THEN                     043
211           IF c$=CHR$(34) THEN                                     582
212             qmode=qmode XOR Q.BASIC.DOUBLE                        914
213             IF (qmode AND Q.BASIC.DOUBLE)=0 THEN                  533
214               IF (qmode AND Q.BASIC.DATA)=0 THEN                  467
215                 qmode=qmode AND NOT Q.ENCLOSED                    500
216               END IF                                              834
217             END IF                                                160
218             GOTO quit                                             964
219           END IF                                                  658
220           IF qmode AND Q.BASIC.DOUBLE THEN                        159
221             qmode=qmode OR Q.ENCLOSED                             945
222           END IF                                                  727
223           IF qmode AND Q.BASIC.DATA THEN                          403
224             IF (qmode AND Q.BASIC.DOUBLE)=0 AND c$=":"
                 THEN                                                 375
225               qmode=qmode AND NOT (Q.ENCLOSED OR
                   Q.BASIC.DATA)                                      951
226               GOTO quit                                           335
227             END IF                                                541
228           ELSE                                                    086
229             qmode=qmode OR Q.ENCLOSED                             028
230           END IF                                                  578
231           IF (qmode AND (Q.BASIC.DOUBLE OR
               Q.BASIC.DATA))=0 THEN                                  140
232             IF c$=" " THEN                                        794
233               IF qmode AND Q.BASIC.NONWHITE THEN                  952
234                 qmode=qmode AND NOT Q.BASIC.NONWHITE              104
```

```
235                ELSE                                     441
236                  qmode=qmode OR Q.IGNORE                710
237                END IF                                   796
238              ELSE                                       453
239                qmode=qmode AND NOT Q.IGNORE OR
                    Q.BASIC.NONWHITE                        401
240                IF c$="'" THEN                           102
241                  qmode=qmode OR Q.BASIC.REM OR Q.IGNORE 675
242                ELSEIF (qmode AND Q.BASIC.PRECSYM)=0 THEN 383
243                  CALL isbasicsym(MID$(buffer$,i+3,1),flag) 118
244                  IF UCASE$(MID$(buffer$,i,3))="REM" AND
                      NOT flag THEN                         186
245                    qmode=qmode OR Q.BASIC.REM OR
                        Q.DISCARD2 OR Q.IGNORE              383
246                  ELSE                                   192
247                    CALL
                        isbasicsym(MID$(buffer$,i+4,1),flag) 144
248                    IF UCASE$(MID$(buffer$,i,4))="DATA" AND
                        NOT flag THEN                       615
249                      qmode=qmode OR Q.BASIC.DATA OR
                          Q.DISCARD3                        823
250                    END IF                               709
251                  END IF                                 075
252                END IF                                   545
253              END IF' "else c == ' '                     858
254            END IF' "if no modes on                      447
255          quit:                                          772
256          qmode=qmode AND NOT Q.BASIC.PRECSYM            795
257          CALL isbasicsym(c$,flag)                       612
258          IF flag THEN qmode=qmode OR Q.BASIC.PRECSYM    944
259          END IF                                         054
260        ELSEIF lmode=L.ASSEM THEN                        930
261          IF c$=CHR$(10) THEN                            411
262            qmode=0                                      114
263          ELSE                                           379
264            IF (qmode AND Q.ASSEM.MAINMODES)=Q.ASSEM.SINGLE
                THEN                                        187
265              IF c$="'" THEN                             232
266                IF MID$(buffer$,i+1,1)="'" THEN          733
267                  qmode=qmode OR Q.DISCARD1 OR Q.ENCLOSED 502
268                ELSE                                     319
269                  qmode=qmode AND NOT (Q.ENCLOSED OR
                      Q.ASSEM.MAINMODES)                    293
270                END IF                                   939
271              ELSE                                       964
272                qmode=qmode OR Q.ENCLOSED                266
273              END IF                                     576
274            ELSEIF (qmode AND
                Q.ASSEM.MAINMODES)=Q.ASSEM.DOUBLE THEN      614
275              IF c$=CHR$(34) THEN                        093
276                qmode=qmode AND NOT (Q.ENCLOSED OR
                    Q.ASSEM.MAINMODES)                      931
277              ELSE                                       804
278                qmode=qmode OR Q.ENCLOSED                210
279              END IF                                     456
280            ELSEIF (qmode AND
                Q.ASSEM.MAINMODES)<>Q.ASSEM.COMMENT THEN    731
281              IF c$=" " THEN                             785
282                IF qmode AND Q.ASSEM.NONWHITE THEN       022
283                  qmode=qmode AND NOT Q.ASSEM.NONWHITE   209
284                ELSE                                     074
285                  qmode=qmode OR Q.IGNORE                571
286                END IF                                   145
287              ELSE                                       034
288                qmode=qmode AND NOT Q.IGNORE OR
                    Q.ASSEM.NONWHITE                        961
289                                                         961
290                IF c$="'" THEN                           790
291                  qmode=qmode OR Q.ASSEM.SINGLE          384
292                ELSEIF c$=CHR$(34) THEN                  972
293                  qmode=qmode OR Q.ASSEM.DOUBLE          383
294                ELSEIF c$="*" AND (qmode AND
                    Q.ASSEM.NOTFIRST)=0 OR c$=";"           814
295 THEN                                                    841
296                  qmode=qmode OR Q.ASSEM.COMMENT OR
                      Q.IGNORE                              428
297                END IF                                   850
298              END IF                                     072
299            END IF                                       526
300            qmode=qmode OR Q.ASSEM.NOTFIRST              011
301          END IF                                         993
302        END IF' "switch lmode                            002
303      END IF' "else qmode and Q.DISCARD                  007
304                                                         007
305      IF (qmode AND Q.IGNORE)=0 and c$<>chr$(10) then    634
306        checksum=checksum*13+ASC(c$) AND &H7FF           429
307      END IF                                             523
308    NEXT                                                 274
309                                                         274
310    ' "won't handle trailing spaces                      553
311                                                         553
312    checksum=checksum MOD SUMRANGE                       754
313                                                         754
314    IF givensum<>-1 AND checksum<>givensum THEN          176
315      PRINT "line "nline": checksum error                604
316      checksum=givensum                                  606
317    END IF                                               988
318 WEND                                                    006
319                                                         006
320 ' "alle Modi müssen abgeschaltet sein!                  117
321                                                         117
322 IF qmode<>0 THEN PRINT "line "nline": unexpected end of
     file                                                   659
323                                                         659
324 CLOSE 2,1                                               600
325                                                         600
326 END                                                     791
327                                                         791
328 ' "Leerzeichen am Zeilenende entfernen                  082
329                                                         082
330 rmtrlspc:                                               679
331                                                         679
332 WHILE RIGHT$(buffer$,1)=" "                             035
333   buffer$=LEFT$(buffer$,LEN(buffer$)-1)                 744
334 WEND                                                    466
335                                                         466
336 RETURN                                                  562
337                                                         562
338 ' "Test auf erlaubtes Zeichen für BASIC-Symbol          201
339                                                         201
340 SUB isbasicsym(c$,flag) STATIC                          816
341                                                         816
342   flag=1                                                960
343                                                         960
344   IF UCASE$(c$)<"A" OR UCASE$(c$)>"Z" THEN IF c$<"0" OR
       c$>"9" THEN IF                                       865
345 c$<>"." THEN flag=0                                     014
346                                                         014
347 END SUB                                                 059
```

Für BASIC-Fans ist diese Version von checksum gedacht.

Sprache: C

Auf der Leserdiskette befindet sich nur das lauffähige C-Compilat!

ABC Eingabe mit beliebigem Editor (ohne Zeilennummern)
Überprüfen mit checksum (alte Version)

```
 1 /*                                                       653
 2 ** CheckSum.c                                            653
 3 ** Copyright (C) 1988, CW-Publikationen, München         653
 4 ** all rights reserved - alle Rechte vorbehalten         653
 5 **                                                       653
 6 ** 15-Aug-1987 created                                   653
 7 ** 17-Jun-1988 added L_ASSEM                             653
 8 */                                                       358
 9                                                          358
10 #include <ctype.h>                                       034
11 #include <stdio.h>                                       688
12 #ifndef AZTEC_C                                          112
13 #include <stdlib.h>                                      505
14 #include <string.h>                                      443
15 #else                                                    019
16 char *strchr();                                          553
17 #endif                                                   106
18 #include "checklib.h"                                    084
19                                                          084
20 /*** Summen prüfen und Quelltext erzeugen ***/           274
21                                                          274
22 static void chksrc __ARGS((FILE *, FILE *, int, int,
    int)); /* added 17-                                     296
23 Jun-88 */                                                465
24                                                          465
25 static void chksrc(filein, fileout, tabstart, tabstep,
    lmode)                                                  766
26 FILE *filein;                                            658
27 FILE *fileout;                                           019
28 int tabstart;                                            846
29 int tabstep;                                             579
30 int lmode;                                               174
31  {                                                       337
32  int line;      /* Zeilenzähler */                       149
33  int qmode;                                              201
34  int checksum; /* Prüfsumme */                           344
35  int givensum; /* angegebene Summe */                    554
36  char *s, *t;                                            406
37                                                          406
38  char buffer[MAXBUFFER]; /* Zeilenpuffer */              703
39                                                          703
40  qmode    = 0;                                           933
41  checksum = 0;                                           044
42  line     = 1;                                           981
43                                                          981
44  while(fgets(buffer, MAXBUFFER, filein) != NULL)         287
45   {                                                      806
46   if((s = strchr(buffer, '\n')) != NULL)                 249
47    *s = '\0';                                            684
48   if(!detab(tabstart, tabstep, buffer, MAXBUFFER))       472
49    {                                                     115
50    rmtrlspc(buffer);                                     648
51    if(strlen(buffer) < SUMWIDTH)                         584
52     goto nosum; /* GOTO? aaarrggghh! */                  660
53    s = buffer + strlen(buffer) - SUMWIDTH;               885
54    givensum = 0;                                         155
55    for(t = s; *t != '\0'; ++t)                           585
56     if(isdigit(*t))                                      693
57      givensum = givensum * 10 + *t - '0';                304
58     else                                                 153
59      {                                                   064
60      nosum:                                              004
61      givensum = -1;                                      492
62      doerror(line, NO_SUM);                              810
63      break;                                              026
64      }                                                   463
65    if(givensum != -1)                                    781
66     {                                                    036
67     *s = '\0';                                           623
68     rmtrlspc(buffer);                                    876
69     }                                                    273
70    strcat(buffer, "\n");                                 738
71    fprintf(fileout, "%s", buffer);                       273
```

```
 72      qmode = strip(qmode, lmode, buffer);                    423
 73      checksum = sum(checksum, buffer);                       733
 74      if(givensum != -1 && checksum != givensum)              070
 75       {                                                      033
 76       doerror(line, BAD_SUM);                                965
 77       checksum = givensum;                                   534
 78       }                                                      923
 79     }                                                        884
 80    else                                                      261
 81      doerror(line, OV_TABS);                                 976
 82    ++line;                                                   029
 83    }                                                         502
 84                                                              502
 85   /* alle Modi müssen abgeschaltet sein! */                  980
 86                                                              980
 87   if(qmode != 0)                                             500
 88     doerror(line, BAD_END);                                  210
 89   }                                                          807
 90                                                              807
 91 /*** Parameter auswerten ***/                                893
 92                                                              893
 93 int main(argc, argv)                                         895
 94 int argc;                                                    986
 95 char *argv[];                                                737
 96   {                                                          512
 97   int ac;                                                    398
 98   char **av;                                                 426
 99   int lmode;                                                 909
100   int tabstart;                                              200
101   int tabstep;                                               437
102   char *srcname;                                             665
103   char *dstname;                                             008
104   FILE *filein;                                              860
105   FILE *fileout;                                             309
106                                                              309
107 /* Modi "undefiniert" */                                     875
108                                                              875
109   lmode    = 0;                                              568
110   tabstart = 0;                                              653
111   tabstep  = 0;                                              940
112   srcname  = NULL;                                           720
113   dstname  = NULL;                                           395
114                                                              395
115   for(ac = argc - 1, av = argv + 1; ac != 0; --ac, ++av)     118
116    {                                                         657
117    if((*av)[0] == '-')                                       224
118     {                                                        987
119     switch((*av)[1])                                         789
120      {                                                       140
121      case 'l': /* language */                                210
122       if(lmode == 0)                                         561
123        if(stricmp(*av + 2, "c") == 0)                        107
124         lmode = L_C;                                         962
125        else if(stricmp(*av + 2, "basic") == 0)               960
126         lmode = L_BASIC;                                     266
127        else if(stricmp(*av + 2, "assem") == 0) /* added
            17-Jun-88 */                                         233
128         lmode = L_ASSEM;                                     038
129        else                                                  007
130         fprintf(stderr, "language \"%s\" not implemented,
             ignored\n", *av
131 + 2);                                                        232
132       else                                                   941
133        fprintf(stderr, "extra language ignored\n");          078
134       break;                                                 613
135      case 'o': /* output */                                  957
136       if(dstname == NULL)                                    138
137        dstname = *av + 2;                                    515
138       else                                                   181
139        fprintf(stderr, "extra output file ignored\n");       470
140       break;                                                 062
141      case 't': /* tabulator start */                         910
142       if(tabstart == 0)                                      684
143        {                                                     199
144        if((tabstart = atoi(*av + 2)) < MINSTART ||           662
         tabstart > MAXSTART)                                    922
145         {                                                    869
146         fprintf(stderr, "bad tabstart specifier %d
          ignored\n", tabstart);                                 044
147         tabstart = 0;                                        177
148         }                                                    378
149        }                                                     943
150       else                                                   904
151        fprintf(stderr, "extra tabstart ignored\n");          388
152       break;                                                 804
153      case 's': /* tabulator step */                          853
154       if(tabstep == 0)                                       182
155        {                                                     441
156        if((tabstep = atoi(*av + 2)) < MINSTEP || tabstep >
         MAXSTEP)                                                013
157         {                                                    292
158         fprintf(stderr, "bad tabstep specifier %d
          ignored\n", tabstep);                                  579
159         tabstep = 0;                                         618
160         }                                                    015
161        }                                                     320
162       else                                                   425
163        fprintf(stderr, "extra tabstep ignored\n");           503
164       break;                                                 063
165      }                                                       944
166     }                                                        109
167    else                                                      270
168     {                                                        585
169     if(srcname == NULL)                                      427
170      srcname = *av;                                          009
171     else                                                     130
172      fprintf(stderr, "extra source file \"%s\" ignored\n",
       *av);                                                     567
173     }                                                        352
174    }                                                         605
175                                                              605
176 /* nicht definierte Modi mit Defaults belegen */             787
177                                                              787
178   if(lmode == 0)                                             074
179    lmode = DEFAULT_MODE;                                     132
180   if(tabstart == 0)                                          703
181    tabstart = DEFAULT_START;                                 614
182   if(tabstep == 0)                                           007
183    tabstep = DEFAULT_STEP;                                   502
184                                                              502
185 /* Files öffnen, Übergabe */                                 980
186                                                              980
187   if(srcname != NULL)                                        558
188    {                                                         233
189    if((filein = fopen(srcname, "r")) != NULL)                459
190     {                                                        994
191     if((fileout = dstname != NULL? fopen(dstname, "w"):
      stdout) != NULL)                                           470
192      {                                                       089
193      chksrc(filein, fileout, tabstart, tabstep, lmode);      399
194                                                              399
195      if(dstname != NULL)                                     288
196       fclose(fileout);                                       468
197      }                                                       065
198     else                                                     058
199      fprintf(stderr, "cannot open for output\n");            286
200     fclose(filein);                                          241
201     }                                                        210
202    else                                                      347
203     fprintf(stderr, "cannot open for input\n");              382
204    }                                                         995
205   else                                                       228
206    fprintf(stderr, "no filename specified\n");               693
207                                                              693
208   return 0;                                                  404
209   }                                                          281
```

Checksum als C-Source für Do-it-yourself-Compilierer.

ABC Eingabe mit beliebigem Editor (ohne Zeilennummern) Überprüfen mit checksum (alte Version)

```
 1 /*                                                             653
 2 ** CheckLib.c - common subroutines                             653
 3 ** Copyright (C) 1988, CW-Publikationen, München               653
 4 ** all rights reserved - alle Rechte vorbehalten               653
 5 **                                                             653
 6 ** 15-Aug-1987 created                                         653
 7 ** 17-Jun-1988 improved L_C, fixed L_BASIC, added L_ASSEM      653
 8 */                                                             358
 9                                                                358
10 #include <ctype.h>                                             034
11 #include <stdio.h>                                             688
12 #ifndef AZTEC_C                                                112
13 #include <string.h>                                            446
14 #endif                                                         279
15 #include "checklib.h"                                          877
16                                                                877
17 /*** "stricmp()" und "strnicmp()" für Manx Aztec C ***/         315
18                                                                315
19 #ifdef AZTEC_C                                                 149
20                                                                149
21 int strnicmp(s, t, n)                                          314
22 char *s;                                                       028
23 char *t;                                                       795
24 int n;                                                         899
25  {                                                             570
26  while(--n >= 0 && *s != '\0' && toupper(*s) ==
     toupper(*t))                                                 389
27   {                                                            084
28   ++s;                                                         808
29   ++t;                                                         329
30   }                                                            306
31                                                                306
32  return n < 0 || *s == '\0' && *t == '\0'? 0:                  064
33         *s == '\0'? -1: *t == '\0'? 1: toupper(*s) -
             toupper(*t);                                         387
34  }                                                             060
35                                                                060
36 int stricmp(s, t)                                              013
37 char *s;                                                       287
38 char *t;                                                       150
39  {                                                             025
40  while(*s != '\0' && toupper(*s) == toupper(*t))               009
41   {                                                            240
42   ++s;                                                         876
43   ++t;                                                         973
44   }                                                            486
45                                                                486
46  return *s == '\0' && *t == '\0'? 0:                           682
47         *s == '\0'? -1: *t == '\0'? 1: toupper(*s) -
             toupper(*t);                                         653
48  }                                                             422
49                                                                422
50 #endif                                                         935
51                                                                935
52 /*** Leerzeichen am Zeilenende entfernen ***/                   021
```

```
 53                                                             021
 54 void rmtrlspc(buffer)                                        381
 55 char *buffer;                                                146
 56  {                                                           021
 57  char *s;                                                    551
 58  bool termlf;                                                564
 59                                                              564
 60  if(termlf = ((s = buffer + strlen(buffer) - 1) >= buffer
      && *s == '\n'))                                            395
 61   --s;                                                       691
 62                                                              691
 63  while(s >= buffer && *s == ' ')                             702
 64   --s;                                                       430
 65                                                              430
 66  if(termlf)                                                  428
 67   *++s = '\n';                                               039
 68                                                              039
 69  *(s + 1) = '\0';                                            691
 70  }                                                           916
 71                                                              916
 72 /*** Tabulatoren durch Leerzeichen ersetzen ***/              082
 73                                                              082
 74 bool detab(start, step, buffer, size)                        274
 75 int start;                                                   914
 76 int step;                                                    076
 77 char *buffer;                                                573
 78 int size;                                                    962
 79  {                                                           341
 80  char *lim, *end, *s;                                        068
 81  int pos;                                                    380
 82  int ins;                                                    272
 83                                                              272
 84  lim = buffer + size;                                        558
 85  end = buffer + strlen(buffer); /* points to '\0' */         907
 86                                                              907
 87  for(pos = 1; *buffer != '\0'; ++pos)                        857
 88   if(*buffer == '\t')                                        053
 89    {                                                         812
 90    ins = (pos < start? start - pos: step - (pos - start)
       % step) - 1;                                              511
 91    if(end + ins >= lim)                                      828
 92     return TRUE; /* kein Platz mehr! */                      085
 93                                                              085
 94    if(ins != 0)                                              565
 95     {                                                        324
 96     for(s = end; s > buffer; --s)                            437
 97      *(s + ins) = *s;                                        313
 98     end += ins;                                              309
 99     pos += ins;                                              904
100     }                                                        637
101                                                              637
102    while(ins >= 0)                                           612
103     {                                                        935
104     *buffer++ = ' ';                                         475
105     --ins;                                                   998
106     }                                                        811
107    }                                                         428
108   else                                                       725
109    ++buffer;                                                 412
110                                                              412
111  return FALSE; /* kein Fehler */                             780
112  }                                                           025
113                                                              025
114 /*** Test auf erlaubtes Zeichen für BASIC-Symbol ***/         679
115                                                              679
116 static bool isbasicsym __ARGS((char)); /* added 17-Jun-88
    */                                                           962
117                                                              962
118 static bool isbasicsym(c)                                    457
119 char c;                                                      357
120  {                                                           668
121  return isalpha(c) || isdigit(c) || c == '.';                835
122  }                                                           740
123                                                              740
124 /*** Modus-Handling ***/                                      138
125                                                              138
126 #define Q_IGNORE 0002                                        270
127                                                              270
128 #define Q_DISCARD  0014                                      283
129 #define Q_DISCARD1 0004                                      308
130 #define Q_DISCARD2 0010                                      975
131 #define Q_DISCARD3 0014                                      023
132                                                              023
133 #define Q_C_MAINMODES 0060                                   594
134 #define Q_C_SINGLE    0020                                   522
135 #define Q_C_DOUBLE    0040                                   753
136 #define Q_C_COMMENT   0060                                   766
137 #define Q_C_NONWHITE  0100                                   707
138                                                              707
139 #define Q_BASIC_DOUBLE    0020                               963
140 #define Q_BASIC_DATA      0040                               292
141 #define Q_BASIC_REM       0100                               328
142 #define Q_BASIC_NONWHITE 0200                                069
143 #define Q_BASIC_PRECSYM   0400 /* added 17-Jun-88 */         501
144                                                              501
145 #define Q_ASSEM_MAINMODES 0060 /* added 17-Jun-88 */         908
146 #define Q_ASSEM_SINGLE    0020                               378
147 #define Q_ASSEM_DOUBLE    0040                               631
148 #define Q_ASSEM_COMMENT   0060                               138
149 #define Q_ASSEM_NONWHITE  0100                               829
150 #define Q_ASSEM_NOTFIRST  0200                               302
151                                                              302
152 /*                                                           491
153  * BASIC-DATA-Zeilen mit Datenelementen,                     491
154  * die nicht mit einem Anführungszeichen                     491
155  * beginnen, aber eine ungerade Anzahl                       491
156  * desselben enthalten, werden manchmal                      491
157  * nicht völlig korrekt behandelt.                           491
158  */                                                          652
159                                                              652
160 int enclosed(qmode, lmode, buffer)                           250
161 int qmode;                                                   278
162 int lmode;                                                   857
163 char *buffer;                                                766
164  {                                                           889
165  char c;                                                     981
166                                                              981
167  if(qmode & Q_DISCARD) /* Ausgleich "look-ahead" o. ä. */    777
168   {                                                          032
169   qmode = qmode & ~Q_DISCARD | (qmode & Q_DISCARD) -
     Q_DISCARD1;                                                 253
170   }                                                          366
171  else                                                        415
172   {                                                          422
173   c = *buffer;                                               121
174                                                              121
175   switch(lmode)                                              293
176    {                                                         884
177    case L_C:                                                 028
178     switch(qmode & Q_C_MAINMODES)                            506
179      {                                                       557
180      case Q_C_SINGLE:                                        899
181       if(c == '\'')                                          745
182        qmode &= ~(Q_ENCLOSED | Q_C_MAINMODES);               004
183       else                                                   685
184        {                                                     836
185        qmode |= Q_ENCLOSED;                                  703
186        if(c == '\\')                                         050
187         qmode |= Q_DISCARD1;                                 027
188        }                                                     476
189       break;                                                 700
190      case Q_C_DOUBLE:                                        847
191       if(c == '"')                                           924
192        qmode &= ~(Q_ENCLOSED | Q_C_MAINMODES);               755
193       else                                                   244
194        {                                                     247
195        qmode |= Q_ENCLOSED;                                  762
196        if(c == '\\')                                         385
197         qmode |= Q_DISCARD1;                                 666
198        }                                                     591
199       break;                                                 959
200      case Q_C_COMMENT:                                       290
201       if(strncmp(buffer, "*/", 2) == 0)                      505
202        qmode = qmode & ~Q_C_MAINMODES | (Q_DISCARD1 |
           Q_IGNORE); /*                                         749
203 changed 17-Jun-88 */                                         198
204       else if(c == '\n')                                     976
205        qmode &= ~Q_IGNORE;                                   726
206       else                                                   415
207        qmode |= Q_IGNORE;                                    009
208       break;                                                 809
209      default:                                                828
210       if(c == ' ')                                           551
211        if(qmode & Q_C_NONWHITE)                              897
212         qmode &= ~Q_C_NONWHITE;                              485
213        else                                                  542
214         qmode |= Q_IGNORE;                                   392
215       else                                                   625
216        {                                                     056
217        qmode = qmode & ~Q_IGNORE | Q_C_NONWHITE;             894
218                                                              894
219        switch(c)                                             572
220         {                                                    415
221         case '\n':                                           969
222          qmode &= ~(Q_IGNORE | Q_C_NONWHITE);                394
223          break;                                              890
224         case '\'':                                           221
225          qmode |= Q_C_SINGLE;                                947
226          break;                                              283
227         case '"':                                            397
228          qmode |= Q_C_DOUBLE;                                736
229          break; /* added 17-Jun-88 */                        774
230         default:                                             721
231          if(strncmp(buffer, "/*", 2) == 0)                   456
232           qmode |= Q_C_COMMENT | Q_DISCARD1 | Q_IGNORE;
              /* changed 17-                                     747
233 Jun-88 */                                                    908
234          break;                                              748
235         }                                                    657
236        }                                                     474
237       break;                                                 306
238      }                                                       007
239     break;                                                   463
240    case L_BASIC:                                             622
241     if(c == '\n')                                            799
242      qmode = 0;                                              749
243     else if((qmode & Q_BASIC_REM) == 0) /* changed
     17-Jun-88 */                                                292
244      {                                                       871
245      if(c == '"')                                            028
246       {                                                      487
247       if(((qmode ^= Q_BASIC_DOUBLE) & Q_BASIC_DOUBLE) ==
       0)                                                        806
248        if((qmode & Q_BASIC_DATA) == 0)                       608
249         qmode &= ~Q_ENCLOSED;                                967
250       break;                                                 887
251       }                                                      416
252      if(qmode & Q_BASIC_DOUBLE)                              552
253       qmode |= Q_ENCLOSED;                                   763
254      if(qmode & Q_BASIC_DATA)                                458
255       if((qmode & Q_BASIC_DOUBLE) == 0 && c == ':')          234
256        {                                                     117
257        qmode &= ~(Q_ENCLOSED | Q_BASIC_DATA);                168
258        break;                                                016
259        }                                                     333
260       else                                                   030
261        qmode |= Q_ENCLOSED;                                  721
262      if((qmode & (Q_BASIC_DOUBLE | Q_BASIC_DATA)) == 0)      264
263       {                                                      507
264       if(c == ' ')                                           822
265        if(qmode & Q_BASIC_NONWHITE)                          151
266         qmode &= ~Q_BASIC_NONWHITE;                          418
267        else                                                  787
268         qmode |= Q_IGNORE;                                   669
269       else                                                   598
270        {                                                     753
```

```
271         qmode = qmode & ~Q_IGNORE | Q_BASIC_NONWHITE;          018
272         if(c == '\'')                                          852
273          qmode |= Q_BASIC_REM | Q_IGNORE; /* changed
              17-Jun-88 */                                         245
274         else if((qmode & Q_BASIC_PRECSYM) == 0) /* added
             17-Jun-88 */                                          503
275          {                                                     518
276          if(strnicmp(buffer, "REM", 3) == 0 &&
              !isbasicsym(buffer[3]))                              840
277           qmode |= Q_BASIC_REM | Q_DISCARD2 | Q_IGNORE;        443
278          else if(strnicmp(buffer, "DATA", 4) == 0 &&           054
279 !isbasicsym(buffer[4]))                                        062
280           qmode |= Q_BASIC_DATA | Q_DISCARD3;                  360
281          }                                                     709
282        } /* else c == ' ' */                                   332
283       } /* if no modes on */                                   731
284      }                                                         436
285     qmode = qmode & ~Q_BASIC_PRECSYM | (isbasicsym(c)?
         Q_BASIC_PRECSYM:                                          165
286 0); /* added 17-Jun-88 */                                      715
287     break;                                                     371
288    case L_ASSEM: /* added 17-Jun-88 */                         807
289     if(c == '\n')                                              732
290      qmode = 0;                                                874
291     else                                                       371
292      {                                                         850
293      switch(qmode & Q_ASSEM_MAINMODES)                         494
294       {                                                        401
295       case Q_ASSEM_SINGLE:                                     709
296        if(c == '\'')                                           291
297         if(buffer[1] == '\'')                                  873
298          qmode |= Q_DISCARD1 | Q_ENCLOSED;                     623
299         else                                                   560
300          qmode &= ~(Q_ENCLOSED | Q_ASSEM_MAINMODES);           389
301        else                                                    958
302         qmode |= Q_ENCLOSED;                                   113
303        break;                                                  617
304       case Q_ASSEM_DOUBLE:                                     618
305        if(c == '"')                                            071
306         qmode &= ~(Q_ENCLOSED | Q_ASSEM_MAINMODES);            712
307        else                                                    921
308         qmode |= Q_ENCLOSED;                                   804
309        break;                                                  940
310       case Q_ASSEM_COMMENT:                                    209
311        break;                                                  945
312       default:                                                 220
313        if(c == ' ')                                            431
314         if(qmode & Q_ASSEM_NONWHITE)                           143
315          qmode &= ~Q_ASSEM_NONWHITE;                           241
316         else                                                   002
317          qmode |= Q_IGNORE;                                    548
318        else                                                    693
319         {                                                      940
320         qmode = qmode & ~Q_IGNORE | Q_ASSEM_NONWHITE;          600
321                                                                600
322         switch(c)                                              374
323          {                                                     889
324          case '\'':                                            788
325           qmode |= Q_ASSEM_SINGLE;                             128
326           break;                                               808
327          case '"':                                             342
328           qmode |= Q_ASSEM_DOUBLE;                             743
329           break;                                               407
330          case '*':                                             697
331           if(qmode & Q_ASSEM_NOTFIRST)                         522
332            break;                                              994
333          case ';':                                             393
334           qmode |= Q_ASSEM_COMMENT | Q_IGNORE;                 442
335           break;                                               530
336          }                                                     871
337         }                                                      208
338        break;                                                  272
339       }                                                        613
340      qmode |= Q_ASSEM_NOTFIRST;                                320
341      }                                                         189
342     break;                                                     341
343    } /* switch lmode */                                        884
344   } /* else qmode & Q_DISCARD */                               451
345                                                                451
346  return qmode;                                                 916
347  }                                                             793
348                                                                793
349 /*** irrelevante Zeichen entfernen ***/                        399
350                                                                399
351 int strip(qmode, lmode, buffer)                                670
352 int qmode;                                                     754
353 int lmode;                                                     253
354 char *buffer;                                                  530
355  {                                                             869
356  char c, *to;                                                  358
357                                                                358
358  to = buffer;                                                  271
359                                                                271
360  while((c = *buffer) != '\0')                                  754
361   if(((qmode = enclosed(qmode, lmode, buffer++)) &
       Q_IGNORE) == 0)                                             828
362    *to++ = c;                                                  778
363                                                                778
364  /* won't handle trailing spaces */                            016
365                                                                016
366  *to = '\0';                                                   011
367                                                                011
368  return qmode;                                                 188
369  }                                                             521
370                                                                521
371 /*** Prüfsumme bilden ***/                                     919
372                                                                919
373 int sum(checksum, buffer)                                      441
374 int checksum;                                                  586
375 char *buffer;                                                  067
376  {                                                             994
377  char c;                                                       042
378                                                                042
379  while((c = *buffer++) != '\n')                                781
380   checksum = checksum * 13 + (c & 0377) & 0x7ff; /*
        changed 17-Jun-88 */                                       030
381                                                                030
382  return checksum % SUMRANGE;                                   035
383  }                                                             580
384                                                                580
385 /*** Fehlermeldungen ausgeben ***/                             514
386                                                                514
387 void doerror(line, error)                                      628
388 int line;                                                      778
389 int error;                                                     714
390  {                                                             213
391  char *msg;                                                    367
392                                                                367
393  switch(error)                                                 312
394   {                                                            083
395   case NO_SUM:                                                 006
396    msg = "no checksum";                                        321
397    break;                                                      185
398   case BAD_SUM:                                                542
399    msg = "checksum error";                                     718
400    break;                                                      390
401   case BAD_END:                                                125
402    msg = "unexpected end of file";                             625
403    break;                                                      081
404   case OV_TABS:                                                665
405    msg = "too many tabs";                                      789
406    break;                                                      749
407   case OV_NDNT:                                                767
408    msg = "indent > width";                                     370
409    break;                                                      546
410   case OV_SPC:                                                 618
411    msg = "too many blanks";                                    287
412    break;                                                      015
413   }                                                            320
414  fprintf(stderr, "line %d: %s\n", line, msg);                  313
415  }                                                             098
```

C-Hauptprogramm von checksum (Lattice und Aztec).

ABC

Eingabe mit beliebigem Editor
(ohne Zeilennummern)
Überprüfen mit checksum (alte Version)

```
 1 /*                                                              653
 2 ** CheckLib.h - common constants, subroutines etc.              653
 3 ** Copyright (C) 1988, CW-Publikationen, München                653
 4 ** all rights reserved - alle Rechte vorbehalten                653
 5 **                                                              653
 6 ** 15-Aug-1987 created                                          653
 7 ** 17-Jun-1988 added L_ASSEM                                    653
 8 */                                                              358
 9                                                                 358
10 /*** compiler ***/                                              580
11                                                                 580
12 #ifdef AMIGA                                                    380
13 #define LATTICE_C                                               662
14 #endif                                                          279
15                                                                 279
16 #ifndef __ARGS                                                  857
17 #ifdef AZTEC_C                                                  515
18 #define __ARGS(a) ()                                            127
19 #else                                                           351
20 #define __ARGS(a) a                                             499
21 #endif                                                          332
22 #endif                                                          693
23                                                                 693
24 /*** boolean ***/                                               259
25                                                                 259
26 typedef int bool;                                               698
27                                                                 698
28 #define FALSE 0                                                 875
29 #define TRUE  1                                                 366
30                                                                 366
31 /*** Bereich für Prüfsumme (0 .. SUMRANGE - 1) ***/              692
32                                                                 692
33 #define SUMWIDTH 3                                              694
34 #define SUMRANGE 1000                                           099
35                                                                 099
36 /*** unterstützte Sprachen (lmode) ***/                          657
37                                                                 657
38 #define L_C      1                                              886
39 #define L_BASIC  2                                              839
40 #define L_ASSEM  3                                              036
41 #define L_PASCAL 4                                              269
42                                                                 269
43 /*** global definierte Bitpositionen (qmode) ***/               219
44                                                                 219
45 #define Q_ENCLOSED 0001                                         259
```

```
46                                                              259
47 /*** Defaults etc. ***/                                       281
48                                                              281
49 #define DEFAULT_MODE L_C                                     218
50                                                              218
51 #define MINSTART        1                                    575
52 #define MAXSTART       64                                    603
53 #define DEFAULT_START 1                                      684
54                                                              684
55 #define MINSTEP        1                                     743
56 #define MAXSTEP       32                                     700
57 #define DEFAULT_STEP 8                                       910
58                                                              910
59 #define MAXBUFFER 512                                        992
60                                                              992
61 /*** Fehlercodes für "doerror()" ***/                          838
62                                                              838
63 #define NO_SUM  1                                            166
64 #define BAD_SUM 2                                            365
65 #define BAD_END 3                                            599
66 #define OV_TABS 4                                            987
67 #define OV_NDNT 5                                            210
68 #define OV_SPC  6                                            528
69                                                              528
70 /*** globale Funktionen ***/                                  142
71                                                              142
72 void rmtrlspc __ARGS((char * /*buffer*/));                   069
73 bool detab __ARGS((int /*start*/, int /*step*/, char *
    /*buffer*/,                                                 076
74  int /*size*/));                                             238
75 int enclosed __ARGS((int /*qmode*/, int /*lmode*/, char *
    /*buffer*/));                                               250
76 int strip __ARGS((int /*qmode*/, int /*lmode*/, char *
    /*buffer*/));                                               343
77 int sum __ARGS((int /*checksum*/, char * /*buffer*/));       106
78 void doerror __ARGS((int /*line*/, int /*error*/));          183
```

Headerfile von checkLib.c

ABC **Eingabe mit beliebigem Editor (ohne Zeilennummern)**
Überprüfen mit checksum (alte Version)

```
 1 #                                                  035
 2 # CheckSum.make                                    813
 3 #                                                  364
 4                                                    364
 5 .include : rules.make                              883
 6                                                    883
 7 checksum : checksum.o checklib.o                   482
 8  $(LINK2)                                          719
 9                                                    719
10 checksum.o : checksum.c checklib.h                 392
11 checklib.o : checklib.c checklib.h                 621
```

Montageanweisungen für checksum.

ABC **Eingabe mit beliebigem Editor (ohne Zeilennummern)**
Überprüfen mit checksum (alte Version)

```
 1 #                                                  035
 2 # Rules.make                                       973
 3 #                                                  396
 4                                                    396
 5 .silent :                                          943
 6                                                    943
 7 .start :                                           941
 8  FailAt 1                                          143
 9                                                    143
10 .end :                                             574
11  FailAt 10                                         112
12                                                    112
13 .c.o :                                             160
14  LC1 -b1 -ceft -w -oT: $*                          103
15  LC2 -r1 -v -o$*.o T:$*                            250
16                                                    250
17 LINK2 = BLink LIB:c.o+$1+$2 TO $$ LIB
   LIB:lcs.lib+LIB:amiga.lib SC SD ND                 802
```

Regeln für Lattice-C-4.01

Welche Preferences hätten'S denn gern?

Für einige Anwendungen ist es sinnvoll, manchmal notwendig, die durch „Preferences" festgelegten Voreinstellungen zu verändern und mehrere Varianten abrufbereit zu halten.

Schon kleine Änderungen der Preferences sind aufwendig, da mehrere Auswahlschirme durchwandert werden müssen. Auch die fehlende Digitalanzeige an den Schiebereglern, auf Neudeutsch auch Propgadgets genannt, so zur Farbeinstellung, läßt die Umstellung meist zum Probierverfahren werden.

Vielmehr wäre es wünschenswert, verschiedene Preferences, entsprechend der im DEVS:-Verzeichnis abgelegten „system-configuration", frei zur Auswahl zu haben.

Dies kann durch Neubenennung der genannten Datei erfolgen, ebenso kann das mittels ASSIGN festgelegte Verzeichnis „DEVS:" umgelenkt werden. Von Komfort keine Spur, aber es gibt noch einen anderen Weg.

Einer der ersten Lesezugriffe auf die Diskette in DF0: — für die Benutzer von Kickstart 1.3 durch Autoboot natürlich auch jedes andere entsprechend vorbereitete Device — gilt der Datei „DEVS:system-configuration". Sie enthält die „struct Preferences", eine Sammlung all jener Daten, die sich anhand des Programms „Preferences" einstellen lassen — und sogar noch einiger weiterer mehr. Ebenfalls von „Preferences" wird diese Datei auch erstmalig erzeugt, sofern sie noch nicht existiert, dies jedoch immer unter dem genannten Namen. Wollen Sie also eine bereits erzeugte Konfiguration als alternativen Datensatz erhalten, so kopieren Sie ihn einfach unter einem neuen Namen am besten ins gleiche Verzeichnis, etwa durch ...

```
Copy DEVS:system-configuration DEVS:alternate
```

Nun kann die ursprüngliche Voreinstellung weiter durch „Preferences" verändert werden und auf diese Weise eine ganze Reihe alternativer Preferences erzeugt werden. Diese müssen jetzt nur noch in irgendeiner Weise wieder geladen werden. Dies geschieht durch den neuen Befehl „SetPrefs", dem als Parameter der Name der neuen Preferences-Datei mitgegeben werden muß.

Ist der Name ohne zugehörigen Gerätenamen spezifiziert worden, erkennbar am fehlenden eingeschlossenen Doppelpunkt, so wird die Datei relativ zu „DEVS:" gesucht, andernfalls wird der Name genau beachtet. Bedenken Sie, daß manche Einstellungen, so etwa die Auswahl von INTERLACE für den Workbench-Schirm, nur beim Bootvorgang beachtet werden und durch „SetPrefs" keinerlei Auswirkung zeigen.

Das Programm ist lauffähig unter Kickstart 1.2 und hoffentlich auch höheren Versionen. Es kann sowohl mit Aztec-C als auch mit Lattice-C compiliert werden, jedoch muß es sich bei den Header-Files zumindest um die zwischenzeitlich in Umlauf geratenen Versionen 1.2.1 handeln, 1.3-Versionen schaden natürlich auch nicht. Da unter Kickstart 1.2 die Preferences noch von konstanter Größe sind, muß im Programm abgefragt werden, von welcher Version denn die aktuellen Bibliotheken, in diesem Fall nur Intuition, sind.

Das letzte Element der 1.2-Preferences gibt im Falle von Kickstart 1.3 und höheren Versionen dann an, um wieviele 64-Bytes-Blöcke die Struktur erweitert wurde.

Das Argument „SAVED" bewirkt gleich dem Preferences-Schalter „LAST SAVED", daß die zuletzt gespeicherte Konfiguration geladen wird. „RESET" stellt entsprechend die Ersatzwerte ein, die auch dann verwendet werden, wenn beim Bootvorgang keine Konfigurationsdatei gefunden wird („Default-City").

Übrigens besitzt die Preferences-Struktur ebensowenig wie das zugehörige File eine spezielle magische Kennung, im Gegensatz etwa zu den Icons, so daß Sie darauf achten sollten, daß es sich bei der Datei des spezifizierten Namens auch wirklich um eine „system-configuration" handelt — andernfalls könnten Sie sich nämlich Ihr eigenes Guru-Grab schaufeln.

(Ralph Babel)

Programmname: SetPrefs
Sprache: C, Hex-Compilat

Auf Leserdiskette

HEX Hexdump des lauffähigen Compilats. Eingabe mit Nanomon

```
000 00 00 03 f3 00 00 00 00 00 00 00 03 00 00 00 00 ad
010 00 00 00 02 00 00 01 b2 00 00 00 9a 00 00 00 01 99
020 00 00 03 e9 00 00 01 b2 23 cf 00 00 00 1c 23 c0 d2
030 00 00 00 24 23 c8 00 00 00 28 2c 79 00 00 00 04 b3
040 23 ce 00 00 00 04 93 c9 4e ae fe da 28 40 4a ac 5f
050 00 ac 67 00 00 f6 61 00 01 8c 91 c8 20 2c 00 ac f9
060 e5 88 20 30 08 10 e5 88 48 e7 00 30 45 f9 00 00 51
070 00 ac 47 f9 00 00 00 2c 20 40 70 00 10 18 42 30 99
080 08 00 26 c8 20 39 00 00 00 24 20 79 00 00 00 28 9c
090 43 f0 08 00 0c 21 00 20 52 c8 ff fa 42 29 00 01 0c
0a0 12 18 67 5c 0c 01 00 20 67 f6 0c 01 00 09 67 f0 92
0b0 26 ca 0c 01 00 22 67 14 14 c1 12 18 67 42 0c 01 06
0c0 00 20 67 04 14 c1 60 f2 42 1a 60 d4 12 18 67 30 4f
0d0 0c 01 00 22 67 f2 0c 01 00 2a 66 20 12 18 0c 01 59
0e0 00 4e 67 06 0c 01 00 6e 66 04 72 0a 60 0e 0c 01 89
0f0 00 45 67 06 0c 01 00 65 66 02 72 1b 14 c1 60 cc 49
100 42 12 42 93 20 3c 00 00 00 28 90 8b 46 80 e4 88 2b
110 4c df 0c 00 48 79 00 00 00 2c 2f 00 4e b9 00 00 c4
```

```
120 05 1c 23 c0 00 00 00 10 4e b9 00 00 05 2c 23 c0 a2
130 00 00 00 14 23 c0 00 00 00 18 4e b9 00 00 02 ba ee
140 70 00 2e 79 00 00 00 1c 4e 75 61 00 00 98 61 00 ed
150 00 82 23 c0 00 00 00 20 2f 00 42 a7 24 40 20 2a 4d
160 00 24 67 10 2c 79 00 00 00 08 20 40 22 28 00 00 3d
170 4e ae ff 82 4e b9 00 00 02 ba 70 00 60 04 20 2f cb
180 00 04 2e 79 00 00 00 1c 2f 00 2c 79 00 00 00 04 d4
190 20 39 00 00 00 08 67 06 22 40 4e ae fe 62 4a b9 29
1a0 00 00 00 20 67 0e 4e ae ff 7c 22 79 00 00 00 20 2a
1b0 4e ae fe 86 20 1f 4e 75 48 e7 01 06 2e 3c 00 03 a9
1c0 80 07 2c 78 00 04 4e ae ff 94 4c df 60 80 70 64 2e
1d0 60 b0 41 ec 00 5c 4e ae fe 80 41 ec 00 5c 4e ae aa
1e0 fe 8c 4e 75 43 fa 00 12 70 00 4e ae fd d8 23 c0 5b
1f0 00 00 00 08 67 c2 4e 75 64 6f 73 2e 6c 69 62 72 3d
200 61 72 79 00 4e 55 ff e8 70 3a 3f 00 2f 2d 00 0c c5
210 4e ba 04 5e 5c 4f 2b 40 ff fc 4a 80 66 2e 48 78 48
220 ff fe 48 79 00 00 01 ac 4e ba 03 3a 50 4f 2b 40 66
230 ff f0 4a 80 67 0e 2f 00 4e ba 03 5a 58 4f 2b 40 b9
240 ff ec 60 08 4e ba 03 62 60 00 00 94 48 78 03 ed 16
250 2f 2d 00 0c 4e ba 02 86 50 4f 2b 40 ff e8 4a 80 ca
260 67 58 20 6d 00 08 2f 2d 00 10 2f 08 2f 00 4e ba 46
270 02 9c 4f ef 00 0c 2b 40 ff f8 22 2d 00 10 b0 81 fb
280 66 18 48 78 00 01 2f 01 2f 2d 00 08 4e ba 03 9e 0a
290 4f ef 00 0c 42 ad ff f4 60 14 52 80 66 06 4e ba 0d
2a0 03 08 60 06 20 3c 00 00 00 d4 2b 40 ff f4 2f 2d cf
2b0 ff e8 4e ba 02 44 58 4f 60 08 4e ba 02 ec 2b 40 f6
2c0 ff f4 4a ad ff fc 66 12 2f 2d ff ec 4e ba 02 c6 b2
2d0 58 4f 2f 00 4e ba 02 aa 58 4f 20 2d ff f4 4e 5d 7f
2e0 4e 75 4e 55 fe c4 70 05 2b 40 fe d0 0c ad 00 00 63
2f0 00 02 00 08 66 00 01 58 70 14 2b 40 fe d0 48 78 0b
300 00 21 48 79 00 00 01 b2 4e ba 02 f2 50 4f 23 c0 ce
310 00 00 00 00 4a 80 67 00 01 28 72 0a 2b 41 fe d0 fc
320 22 3c 00 00 00 e8 2f 01 48 6d fe d4 2b 41 fe c4 98
330 4e ba 02 e2 50 4f 20 79 00 00 00 00 0c 68 00 21 43
340 00 14 67 16 10 2d ff bb 02 40 00 ff 22 00 ed 41 34
350 02 81 00 00 ff ff d3 ad fe c4 42 a7 2f 2d fe c4 6a
360 4e ba 02 56 50 4f 2b 40 ff fc 4a 80 67 00 00 b8 a3
370 48 79 00 00 01 c4 20 6d 00 0c 2f 28 00 04 4e ba 30
380 03 34 50 4f 4a 40 66 28 2f 2d fe c4 2f 2d ff fc e6
390 4e ba 02 82 50 4f 48 78 00 01 2f 2d fe c4 2f 2d 46
3a0 ff fc 4e ba 02 88 4f ef 00 0c 42 ad fe d0 60 66 87
3b0 48 79 00 00 01 ca 20 6d 00 0c 2f 28 00 04 4e ba 5a
3c0 02 f4 50 4f 4a 40 66 08 41 f9 00 00 01 d0 60 08 ff
3d0 22 6d 00 0c 20 69 00 04 2f 2d fe c4 2f 08 2f 2d 61
3e0 ff fc 2b 48 fe c8 61 00 fe 1c 4f ef 00 0c 2b 40 29
3f0 fe cc 4a 80 67 1c 2f 2d fe c8 2f 00 20 6d 00 0c 73
400 2f 10 48 79 00 00 01 e6 4e ba 00 b6 4f ef 00 10 9b
410 60 04 42 ad fe d0 2f 2d fe c4 2f 2d ff fc 4e ba b3
420 01 b0 50 4f 60 0c 48 79 00 00 02 04 4e ba 00 92 cf
430 58 4f 2f 39 00 00 00 00 4e ba 01 ae 58 4f 60 20 00
440 48 79 00 00 02 1e 4e ba 00 78 58 4f 60 12 20 6d 37
450 00 0c 2f 10 48 79 00 00 02 42 4e ba 00 64 50 4f dc
460 2f 2d fe d0 4e ba fd 18 58 4f 4e 5d 4e 75 00 00 ae
470 48 e7 00 3a 28 6f 00 14 20 6f 00 18 22 6f 00 1c 5f
480 45 fa 00 3a 4f ef ff 74 26 4f 2c 79 00 00 00 04 10
490 4e ae fd f6 70 ff 4a 1b 57 c8 ff fc 46 80 67 12 f5
4a0 2f 00 48 6f 00 04 48 54 4e b9 00 00 05 00 4f ef 4a
4b0 00 0c 4f ef 00 8c 4c df 5c 00 4e 75 16 c0 4e 75 48
4c0 20 2f 00 04 48 6f 00 08 2f 00 2f 39 00 00 00 14 8d
4d0 4e b9 00 00 04 48 4f ef 00 0c 4e 75 48 e7 20 02 d2
4e0 2c 79 00 00 00 08 4c ef 00 06 00 0c 4e ae ff e2 15
4f0 4c df 40 04 4e 75 00 00 2f 0e 2c 79 00 00 00 08 17
500 22 2f 00 08 4e ae ff dc 2c 5f 4e 75 48 e7 30 02 90
510 2c 79 00 00 00 08 4c ef 00 0e 00 10 4e ae ff d6 15
520 4c df 40 0c 4e 75 00 00 48 e7 30 02 2c 79 00 00 cf
530 00 08 4c ef 00 0e 00 10 4e ae ff d0 4c df 40 0c 9b
540 4e 75 00 00 2f 0e 2c 79 00 00 00 08 4e ae ff ca 5b
550 2c 5f 4e 75 2f 0e 2c 79 00 00 00 08 4e ae ff c4 5d
560 2c 5f 4e 75 48 e7 20 02 2c 79 00 00 00 08 4c ef 32
570 00 06 00 0c 4e ae ff ac 4c df 40 04 4e 75 00 00 17
580 2f 0e 2c 79 00 00 00 08 22 2f 00 08 4e ae ff a6 41
590 2c 5f 4e 75 2f 0e 2c 79 00 00 00 08 22 2f 00 08 af
5a0 4e ae ff 82 2c 5f 4e 75 2f 0e 2c 79 00 00 00 08 dc
5b0 4e ae ff 7c 2c 5f 4e 75 2f 0e 2c 79 00 00 00 04 0f
5c0 4c ef 00 03 00 08 4e ae ff 3a 2c 5f 4e 75 00 00 7e
5d0 2f 0e 2c 79 00 00 00 04 22 6f 00 08 20 2f 00 0c a4
5e0 4e ae ff 2e 2c 5f 4e 75 2f 0e 2c 79 00 00 00 04 38
5f0 22 6f 00 08 4e ae fe 62 2c 5f 4e 75 2f 0e 2c 79 51
600 00 00 00 04 22 6f 00 08 20 2f 00 0c 4e ae fd d8 ab
610 2c 5f 4e 75 2f 0e 2c 79 00 00 00 00 20 6f 00 08 0e
620 20 2f 00 0c 4e ae ff 82 2c 5f 4e 75 2f 0e 2c 79 98
630 00 00 00 00 20 6f 00 08 4c ef 00 03 00 0c 4e ae 32
640 fe bc 2c 5f 4e 75 00 00 4e 55 00 00 20 6d 00 08 1a
650 12 10 b2 2d 00 0d 66 04 20 08 60 10 20 6d 00 08 b4
660 10 10 52 ad 00 08 4a 00 66 e2 70 00 4e 5d 4e 75 cb
670 4e 55 00 00 10 2d 00 0d 02 40 00 ff 3f 00 2f 2d 7a
680 00 08 61 c4 5c 4f 4e 5d 4e 75 4e 55 ff fc 42 ad c9
690 ff fc 20 6d 00 08 4a 10 67 12 12 10 b2 2d 00 0d 24
6a0 66 04 2b 48 ff fc 52 ad 00 08 60 e6 20 2d ff fc 31
6b0 4e 5d 4e 75 20 6f 00 04 22 6f 00 08 70 00 72 00 19
6c0 10 18 12 19 0c 00 00 61 6d 0a 0c 00 00 7a 6e 04 25
6d0 04 00 00 20 0c 01 00 61 6d 0a 0c 01 00 7a 6e 04 55
6e0 04 01 00 20 90 81 66 04 4a 01 66 d4 4e 75 00 00 b4
6f0 00 00 03 ec 00 00 00 06 00 00 00 00 00 00 04 aa 90
700 00 00 04 82 00 00 00 f6 00 00 01 02 00 00 01 4e 75
710 00 00 01 14 00 00 00 2d 00 00 00 01 00 00 05 d8 b4
720 00 00 05 c4 00 00 05 ac 00 00 05 94 00 00 05 84 35
730 00 00 05 70 00 00 05 5c 00 00 05 42 00 00 05 30 68
740 00 00 05 20 00 00 05 06 00 00 04 ea 00 00 04 d4 c1
750 00 00 04 ba 00 00 04 a4 00 00 04 2e 00 00 04 1a 16
760 00 00 04 00 00 00 03 dc 00 00 03 a2 00 00 03 8a 1e
770 00 00 03 4a 00 00 02 dc 00 00 01 fc 00 00 01 c8 5f
780 00 00 01 84 00 00 01 78 00 00 01 6a 00 00 01 5c 2d
790 00 00 01 3e 00 00 01 2c 00 00 01 1c 00 00 01 0e dc
7a0 00 00 01 08 00 00 00 fc 00 00 00 ee 00 00 00 de 5e
7b0 00 00 00 64 00 00 00 5e 00 00 00 4c 00 00 00 46 a2
7c0 00 00 00 1a 00 00 00 0e 00 00 00 08 00 00 00 02 bd
7d0 00 00 00 05 00 00 00 02 00 00 06 08 00 00 05 f0 66
7e0 00 00 04 0c 00 00 03 10 00 00 02 e8 00 00 00 00 9e
7f0 00 00 03 f2 00 00 03 ea 00 00 00 9a 00 21 00 01 00
800 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff 7c
810 ff ff ff ff ff ff ff ff 00 00 00 00 00 00 00 00 cd
820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 82
830 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 df
840 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3c
850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 99
860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f6
870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 53
880 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b0
890 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0d
8a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 6a
8b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 c7
8c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 24
8d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 81
8e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 de
8f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3b
900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 81
910 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 de
920 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3b
930 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 98
940 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f5
950 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 52
960 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 af
970 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0c
980 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 69
990 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 c6
9a0 00 00 00 00 00 00 00 00 44 45 56 53 3a 00 69 6e 9a
9b0 74 75 69 74 69 6f 6e 2e 6c 69 62 72 61 72 79 00 97
9c0 52 45 53 45 54 00 53 41 56 45 44 00 73 79 73 74 26
9d0 65 6d 2d 63 6f 6e 66 69 67 75 72 61 74 69 6f 6e c5
9e0 00 00 25 73 3a 20 45 72 72 6f 72 20 25 6c 64 20 e4
9f0 6c 6f 61 64 69 6e 67 20 22 25 73 22 2e 0a 00 00 8e
a00 49 6e 73 75 66 66 69 63 69 65 6e 74 20 66 72 65 96
a10 65 20 73 74 6f 72 65 2e 0a 00 43 61 6e 6e 6f 74 2c
a20 20 6f 70 65 6e 20 74 68 65 20 49 6e 74 75 69 74 38
a30 69 6f 6e 20 6c 69 62 72 61 72 79 2e 0a 00 55 73 44
a40 61 67 65 3a 20 25 73 20 52 45 53 45 54 20 7c 20 34
a50 53 41 56 45 44 20 7c 20 3c 66 69 6c 65 6e 61 6d 0a
a60 65 3e 0a 00 00 00 03 f2 00 00 03 eb 00 00 00 01 3d
a70 00 00 03 f2                                     6e
```

Lauffähige Hex-Version von SetPrefs

Sprache: C

ABC **Eingabe mit beliebigem Editor (ohne Zeilennummern)**
Überprüfen mit checksum (alte Version)

```
 1 /*                                                      653
 2 ** SetPrefs.c - set preferences                         653
 5 **                                                      653
 6 ** 18-May-1987 created                                  653
 7 ** 09-Apr-1988 enhanced for Kickstart 1.3 & up          653
 8 */                                                      358
 9                                                         358
10 /*** included files (requires at least 1.2.1) ***/      580
11                                                         580
12 #include <exec/types.h>                                 916
13 #include <exec/memory.h>                                196
14 #include <libraries/dos.h>                              441
15 #include <intuition/intuition.h>                        785
16 #include <intuition/intuitionbase.h>                    436
17                                                         436
18 #ifndef AZTEC_C                                         900
19 #include <string.h>                                     330
20 #else                                                   286
21 char *strchr();                                         864
22 #endif                                                  809
23                                                         809
24 /*** external function references ***/                  671
25                                                         671
26 #ifndef AZTEC_C                                         051
27 struct Library *OpenLibrary(char *, ULONG);             571
28 void CloseLibrary(struct Library *);                    792
29 APTR AllocMem(ULONG, ULONG);                            843
30 void FreeMem(APTR, ULONG);                              738
31 void GetDefPrefs(struct Preferences *, ULONG);          093
32 void SetPrefs(struct Preferences *, ULONG, LONG);       261
33 BPTR Open(char *, LONG);                                619
34 LONG Read(BPTR, UBYTE *, LONG);                         738
35 void Close(BPTR);                                       522
36 BPTR Lock(char *, LONG);                                183
```

```
 37 void UnLock(BPTR);                                        617
 38 BPTR CurrentDir(BPTR);                                    855
 39 LONG IoErr(void);                                         274
 40 void printf(char *, );                                    287
 41 void exit(LONG);                                          599
 42 #else                                                     191
 43 struct Library *OpenLibrary();                            558
 44 void CloseLibrary();                                      931
 45 APTR AllocMem();                                          364
 46 void FreeMem();                                           367
 47 void GetDefPrefs();                                       472
 48 void SetPrefs();                                          054
 49 BPTR Open();                                              664
 50 LONG Read();                                              808
 51 void Close();                                             712
 52 BPTR Lock();                                              009
 53 void UnLock();                                            395
 54 BPTR CurrentDir();                                        033
 55 LONG IoErr();                                             150
 56 void printf();                                            367
 57 void exit();                                              023
 58 #endif                                                    336
 59                                                           336
 60 /*** constants ***/                                       974
 61                                                           974
 62 #define ZERO 0L                                           096
 63                                                           096
 64 #define LIBRARY_VERSION_1_2 33                            899
 65                                                           899
 66 #define PREFSIZE_1_2 232L /* sizeof(struct Preferences)
    */                                                        551
 67                                                           551
 68 /*** global symbols (linker & amiga.lib) ***/              637
 69                                                           637
 70 struct IntuitionBase *IntuitionBase;                      795
 71                                                           795
 72 /*** functions ***/                                       177
 73                                                           177
 74 #ifdef AZTEC_C                                            411
 75 static int stricmp(s, t)                                  504
 76 unsigned char *s, *t;                                     741
 77  {                                                        564
 78  int delta;                                               788
 79                                                           788
 80  while((delta = toupper(*s) - toupper(*t)) == 0)          704
 81   {                                                       083
 82   if(*s == '\0')                                          532
 83    {                                                      895
 84    return 0;                                              046
 85    }                                                      723
 86                                                           723
 87   ++s;                                                    511
 88   ++t;                                                    528
 89   }                                                       845
 90                                                           845
 91  return delta;                                            082
 92  }                                                        191
 93 #endif                                                    696
 94                                                           696
 95 #ifndef AZTEC_C                                           152
 96 static LONG LoadPrefs(struct Preferences *, char *,
    ULONG);                                                   838
 97 #endif                                                    071
 98                                                           071
 99 static LONG LoadPrefs(prefs, filename, prefsize)          577
100 struct Preferences *prefs;                                017
101 char *filename;                                           945
102 ULONG prefsize;                                           349
103  {                                                        564
104  char *colonp;                                            370
105  LONG length;                                             551
106  LONG iocode;                                             657
107  BPTR fl, cd;                                             217
108  BPTR fh;                                                 090
109                                                           090
110  if((colonp = strchr(filename, ':')) == NULL)             408
111   {                                                       331
112   if((fl = Lock("DEVS:", ACCESS_READ)) != ZERO)           500
113    {                                                      479
114    cd = CurrentDir(fl);                                   607
115    }                                                      872
116   else                                                    593
117    {                                                      688
118    return IoErr();                                        793
119    }                                                      194
120   }                                                       599
121                                                           599
122  if((fh = Open(filename, MODE_OLDFILE)) != ZERO)          704
123   {                                                       083
124   if((length = Read(fh, (UBYTE *)prefs, prefsize)) ==
    prefsize)                                                 564
125    {                                                      311
126    SetPrefs(prefs, prefsize, TRUE);                       051
127                                                           051
128    iocode = 0;                                            754
129    }                                                      735
130   else                                                    416
131    {                                                      435
132    iocode = length == -1? IoErr():
     ERROR_OBJECT_WRONG_TYPE;                                 256
133    }                                                      405
134   Close(fh);                                              733
135   }                                                       462
136  else                                                     047
137   {                                                       734
138   iocode = IoErr();                                       655
139   }                                                       448
140                                                           448
141  if(colonp == NULL)                                       532
142   {                                                       895
143   UnLock(CurrentDir(cd));                                 085
144   }                                                       230
145                                                           230
146  return iocode;                                           648
147  }                                                        357
148                                                           357
149 void main(argc, argv)                                     346
150 #ifndef AZTEC_C                                           450
151 LONG argc; /* Astartup.obj!!! change to "int"
    otherwise!!! */                                           620
152 #else                                                     608
153 int argc;                                                 743
154 #endif                                                    416
155 char *argv[];                                             423
156  {                                                        526
157  struct Preferences *prefs, P;                            918
158  LONG result, iocode;                                     163
159  char *filename;                                          627
160  ULONG prefsize;                                          807
161                                                           807
162  result = RETURN_WARN;                                    701
163                                                           701
164  if(argc == 2)                                            750
165   {                                                       681
166   result = RETURN_FAIL;                                   203
167                                                           203
168   if((IntuitionBase = (struct IntuitionBase *)            191
169   OpenLibrary("intuition.library",
     (ULONG)LIBRARY_VERSION_1_2)) != NULL)                    077
170    {                                                      124
171    result = RETURN_ERROR;                                 588
172                                                           588
173    GetDefPrefs(&P, prefsize = PREFSIZE_1_2);              483
174                                                           483
175    if(IntuitionBase->LibNode.lib_Version !=
     LIBRARY_VERSION_1_2)                                     128
176     {                                                     787
177     prefsize += P.ext_size * 64;                          223
178     }                                                     976
179                                                           976
180    if((prefs = (struct Preferences *)AllocMem(prefsize,
     0)) != NULL)                                             224
181     {                                                     987
182     if(stricmp(argv[1], "RESET") == 0)                    824
183      {                                                    595
184      GetDefPrefs(prefs, prefsize);                        662
185      SetPrefs(prefs, prefsize, TRUE);                     274
186                                                           274
187      result = RETURN_OK;                                  534
188      }                                                    923
189     else                                                  028
190      {                                                    487
191      filename = stricmp(argv[1], "SAVED") == 0?
       "system-configuration": argv[1];                       205
192                                                           205
193      if((iocode = LoadPrefs(prefs, filename, prefsize))
      != 0)                                                   058
194       {                                                   877
195       printf("%s: Error %ld loading \"%s\".\n", argv[0],
        iocode, filename);                                    695
196       }                                                   968
197      else                                                 177
198       {                                                   376
199       result = RETURN_OK;                                 220
200       }                                                   937
201      }                                                    018
202     FreeMem((APTR)prefs, prefsize);                       879
203     }                                                     312
204    else                                                   265
205     {                                                     520
206     printf("Insufficient free store.\n");                 045
207     }                                                     710
208    CloseLibrary((struct Library *)IntuitionBase);         696
209    }                                                      981
210   else                                                    782
211    {                                                      049
212    printf("Cannot open the Intuition library.\n");        314
213    }                                                      111
214   }                                                       568
215  else                                                     521
216   {                                                       752
217   printf("Usage: %s RESET | SAVED | <filename>\n",
    argv[0]);                                                 729
218   }                                                       410
219                                                           410
220  exit(result);                                            695
221  }                                                        968
```

Voreinstellungen mit diesem C-Listing wählbar machen

```
1 ;                                                         059
2 ;SetPrefsLC.bat                                           962
3 ;                                                         277
4 FailAt 1                                                  759
5 LC1 -b0 -ceft -w -oRAM:t/ SetPrefs                        274
6 LC2 -r1 -s -v RAM:t/SetPrefs                              755
7 BLink LIB:Astartup.obj+RAM:t/SetPrefs.o TO SetPrefs LIB
  LIB:amiga.lib+LIB:lcsnb.lib SC SD ND                      859
8 Delete RAM:t/SetPrefs.o                                   844
9 FailAt 10                                                 254
```

Lattice-Linker zu SetPref

```
1 ;                                                         059
2 ;SetPrefsAC.bat                                           451
3 ;                                                         826
4 FailAt 1                                                  444
5 cc SetPrefs                                               414
6 ln SetPrefs -lc                                           796
7 Delete SetPrefs.o                                         952
8 FailAt 10                                                 482
```

Aztek-Linker zu SetPref

KeyMaps — Zeichen setzen nach Bedarf

Mit „SetMap d-hex" läßt sich die Tastatur Anwender-orientiert organisieren. Wer die interne Verarbeitung von Tastenanschlägen versteht, gelangt leicht zu eigenen Konzepten, die Verrenkungen der Finger überflüssig machen.



Sedizimale Daten, im allgemeinen Sprachgebrauch besser als Hex-Zahlen bekannt, enthalten neben den üblichen Ziffern unseres Zahlensystems sechs weitere in Form der Buchstaben von A bis F. Die Tastaturen der Computer halten sich jedoch an die Normen historischer Rechen- und Schreibmaschinen. Daher waren sedizimale Dateneingaben bisher eher lästige Fingerübungen als flüssige Arbeitsvorgänge.
Die Tastatursoftware des Amiga ist wie das Gesamtkonzept des Rechners flexibel gestaltet. Durch eine sogenannte „KeyMap“ kann nahezu jede denkbare Belegung erzeugt werden, Akzente und andere diakritische Zeichen stellen durch die „Dead Keys“ kein Problem dar. Ferner lassen sich die Tasten sogar mit ganzen Zeichenketten belegen, wobei jede Taste nochmals mit vier „Qualifiers“, das sind unter anderem die Tasten SHIFT, ALT und CTRL, kombiniert und so mit weiteren Funktionen versehen werden kann.
Die Tastatur des Amiga ist ein eigener kleiner Computer, der sich ausschließlich um die Abfrage derselben kümmert und den Hauptprozessor für wichtigere Aufgaben freihält. Dieser Tastaturprozessor, in den meisten Amigas ein Ein-Chip-Abkömmling der 6502, schickt sogenannte „Raw Key Codes“ über die serielle Verbindung zwischen Tastatur und Hauptplatine. Diese rohen Tastencodes geben jedoch lediglich die Lage der Taste auf der Tastatur an und stehen nicht unbedingt in Zusammenhang mit dem Aufdruck auf den Tastenkappen.
Der Empfang dieser Tastaturcodes wird vom „keyboard.device“ übernommen, welches Buch darüber führt, welche Tasten zu welchem Zeitpunkt gedrückt sind. Das „input.device“ verschmilzt diese Daten zusammen mit denen anderer Eingabe-Gerätetreiber, dem „gameport.device“ und dem „timer.device“, zu einem Eingabestrom, der vom „console .device“ und Intuition überwacht und ausgewertet wird.
Andere Programme, etwa solche des Typs „QuickMouse“ oder „PopCLI“, bedienen sich ebenfalls dieser Daten.
Der Eingabestrom besteht aus einer Reihe von sogenannten „Input- Events“, Eingabe-Ereignissen also. Ein Input-Event hat folgende Struktur:

```
struct InputEvent
 {  struct InputEvent *ie_NextEvent;
                                /* das zeitlich n?chste Ereignis */
 UBYTE ie_Class;                /* die Klasse des Ereignisses */
 UBYTE ie_SubClass;             /* weitere Unterteilung der Klasse */
 UWORD ie_Code;                 /* Code des Input-Events */
 UWORD ie_Qualifier;            /* aktuelle Qualifiers zum Zeitpunkt des
                                   Events */
 union
  {
  struct
   {
    WORD ie_X;                  /* Position des Mauszeigers */
   WORD ie_Y;
   },
  APTR ie_EventAddress;
 };
struct timeval ie_TimeStamp
                               /* Zeitpunkt des Ereignisses */
};
```

Diese Struktur entspricht nicht in allen Punkten der Darstellung in <devices/inputevent.h>, sondern wurde in Anlehnung an die „unnamed unions“ in der Hochsprache C++ modifiziert, um anschaulich den Grundgedanken zu demonstrieren.
Eine besondere Rolle spielen dabei die acht Qualifier-Tasten, deren aktueller Zustand immer zusammen mit einem Tastendruck im Eingabestrom übermittelt wird:

```
0: Linke SHIFT-Taste
1: Rechte SHIFT-Taste
2: CAPS LOCK
3: CTRL
4: Linke ALT-Taste
5: Rechte ALT-Taste
6: Linke Amiga-Taste (C=)
7: Rechte Amiga-Taste
```

Mit Ausnahme von CAPS LOCK (siehe unten) sind diese Tasten auch in der Tastaturhardware so angelegt, daß ihre Abfrage uneingeschränkt erfolgen kann. Im Unterschied dazu kann es beim gleichzeitigen Drücken mehrerer anderer (Nicht-Qualifier-) Tasten aufgrund der Struktur der Tastaturmatrix dazu kommen, daß Tasten als gedrückt erkannt werden, die es gar nicht sind. Da dies bei den Qualifier-Tasten, die ja zur gleichzeitigen Betätigung mit anderen Tasten gedacht sind, nicht tragbar ist, wurde hier die Decodierung getrennt vorgenommen.
Alles fängt an mit den Input-Events: Intuition-Programmierer werden gewisse Parallelen zur „IntuiMessage“ erkennen, und tatsächlich existieren hier direkte Entsprechungen: Wird in der Menge der IDCMP-Flags eines Intuition-Fensters die Klasse „RAWKEY“ angegeben, so werden dem Programm (geringfügig aufbereitete) Input-Events in Form von IntuiMessages mit den bereits erwähnten rohen Tastencodes übermittelt. Jedesmal, wenn eine Taste gedrückt oder losgelassen wird, erfolgt Meldung über den UserPort des Windows.
Im Falle der Angabe von „VANILLAKEY“ werden diese rohen Tastencodes in ihre ASCII-Äquivalente gemäß der aktuellen KeyMap umgewandelt. Eine Einschränkung besteht darin, daß nur solche Tasten Beachtung finden, deren Belegung genau ein einzelnes Byte erzeugt: Funktionstasten, Cursortasten und Qualifier bleiben dem Programm also verborgen.
Eine Kombination von KEYKEY und VANILLAKEY resultiert übrigens lediglich in einer Bearbeitung der „VANILLAKEYS“, so daß auf diese Weise keine vollständige Kontrolle der Tastatureingabe möglich ist.
Etwas weitergehend ist eine Abfrage über das „console.device“, das auch vollständig die Anpassung an die aktuelle KeyMap übernimmt. Ihr Programm ist also automatisch ▶

„international“ und funktioniert mit jeder Tastaturbelegung.
Hierzu übergibt man dem console.device vor Eröffnung einen Zeiger auf das Fenster, das als Eingabequelle dienen soll. Wann immer dieses Fenster aktiviert ist, werden alle Tastendrücke in Standard-ASCII-Sequenzen übermittelt. Über die KeyMaps selbst findet sich Wissenswertes im Headerfile. Ein Ausdruck dieser Datei (Pfad: < devices/ keymap.h >) ist ein nützliches Hilfsmittel, wenn man eigene KeyMaps definieren will.
Das Betriebssystem unterhält eine Liste von Keymaps in der „keymap.resource“. Jedes Element dieser Liste enthält eine „struct KeyMap“, die wiederum aus acht Zeigern auf bestimmte Tabellen besteht. Genaugenommen sind nur vier Zeiger notwendig, aber dazu später noch mehr.
Die rohen Tastaturcodes liegen im Bytebereich, wobei die Werte von $00 bis $77 eine gedrückte Taste und $80 bis $F7 das Loslassen der gleichen Tasten bezeichnen. Andere Codes sind nicht benutzt oder dienen der internen Kommunikation mit dem keyboard.device. Von diesen maximal möglichen 120 Tasten werden zur Zeit lediglich 96 Tasten im Falle der deutschen Tastatur (und auch anderer) genutzt. Aus wahrscheinlich historischen Gründen erfolgt noch eine Unterteilung dieser maximal 120 Tasten in sogenannte Low- und High-Keys, wobei sich der niedrige Bereich von 0 bis 63 erstreckt. Bei der Zusammenstellung einer Keymap ist dies jedoch ohne Bedeutung, da alle Tabellen kontinuierlich erstellt werden können und lediglich ab der die High-Keys bezeichenden Position ein Label eingefügt werden muß. Daraus resultieren auch die erwähnten acht Zeiger.

Eingerastete Versalien genießen Sonderrechte

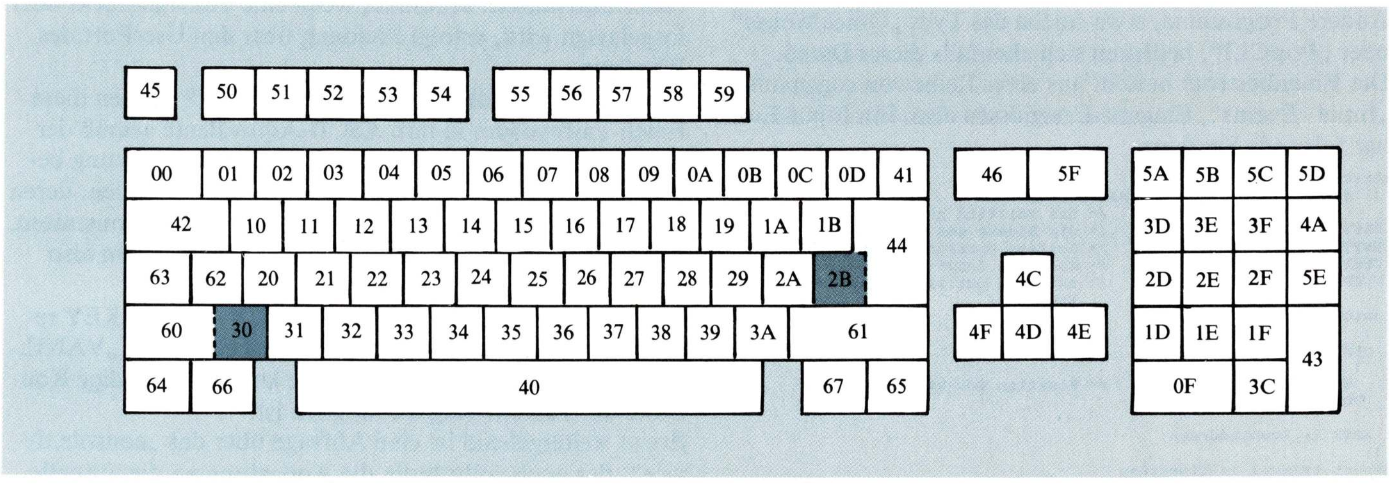
CAPS LOCK nimmt eine Sonderstellung unter den Tasten ein: Sie gilt solange als „gedrückt“, wie die LED leuchtet und der RawCode, der angibt, daß die Taste „losgelassen“ wurde, wird erst beim Erlöschen der LED übermittelt.
CAPS LOCK wirkt genauso wie eine der beiden SHIFT-Tasten, dies jedoch nur bei solchen Tasten, die als „Capsable“ definiert wurden. Auf diese Weise läßt sich beispielsweise eine längere Sequenz an Großbuchstaben eingeben, ohne daß, im Unterschied zur Arretierung der Shift-Taste einer Schreibmaschine, jedesmal bei Betätigung einer Nicht-Buchstabentaste erst CAPS LOCK ausgeschaltet werden muß.
Die Capsable-Tabelle umfaßt 15 Bytes (120 Bits), die ein Bitfeld darstellen. Jedem der rohen Tastencodes von 0 bis 119 entspricht genau ein Bit dieses Feldes. Dabei beginnt die Numerierung im ersten Byte und verläuft innerhalb eines Bytes vom LSB zum MSB. Ist das Bit gelöscht, so wirkt CAPS LOCK auf diese Taste wie SHIFT, andernfalls wird CAPS LOCK ignoriert.

Ständige Wiederholungen werden in einer Tabelle festgehalten

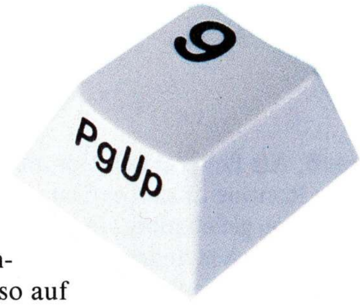
Eine identisch strukturierte Tabelle existiert für die automatische Wiederholfunktion, deren Ansprechzeit und Geschwindigkeit in „Preferences“ festgelegt werden kann. Hier zeigt ein gesetztes Bit an, daß diese Taste nach einer bestimmten Zeitspanne automatisch wiederholt wird.

Typen und Tastencodes

Unter Typen sind weder Männer noch die Anschlagbuchstaben einer Schreibmaschine zu verstehen. Vielmehr gibt es verschiedene Arten von Tasten, die auf unterschiedliche Weise ausgewertet werden. Je nach gesetzten Bits kann festgelegt werden, mit welchen der drei Qualifiers SHIFT (oder CAPS LOCK), ALT und CTRL zusammen welcher Code erzeugt werden soll. Beim Loslassen der Taste in Kombination mit den genannten Qualifiers ließe sich ein separater Code generieren. Ein Verwendungszweck hierfür ist noch niemanden eingefallen, vielleicht hat jedoch ein Leser eine zündende Idee.
Der einfachste Fall eines Tastenereignisses tritt ein, wenn eine Taste immer den gleichen Code erzeugt, unabhängig von eventuellen Qualifiers. Dieser Tastentyp heißt entsprechend „KC_NOQUAL“. Die Tabelle der Tastaturcodes besteht aus vier Bytes für jeden der 120 RawCodes. Im Falle von „KC_NOQUAL“ findet sich im letzten der vier Bytes



So sind die einzelnen Tasten bei Amiga 500 und 2000 mit Steuercodes belegt.



der zu verwendende Code. Soll, etwa bei aktivem SHIFT-Qualifier, ein unterschiedlicher Code generiert werden, muß in der Tabelle der Tastentypen „KCF_SHIFT“ spezifiziert werden. Von den vier Bytes der Code-Tabelle gibt nun das dritte Byte den Tastencode an, der bei gedrückter SHIFT-Taste erzeugt wird, die beiden ersten Bytes sind weiterhin nicht verwendet.
Entsprechendes gilt für die übrigen Kombinationen, eine genaue Übersicht können Sie dem KeyMap-Listing entnehmen. Ist eine Taste übrigens nicht „vorhanden“ (nicht allen rohen Tastencodes entspricht eine Position auf der Tastatur), so wird als Type „KCF_NOP“ spezifiziert. Der Eintrag in der Tabelle der Tastencodes ist dann ohne Bedeutung und wird ignoriert. Doch sind damit noch lange nicht alle Möglichkeiten erschöpft: So lassen sich nicht nur einzelne Bytes erzeugen, es kann jeder Taste auch ein aus mehreren Zeichen bestehender String zugewiesen werden.
Hierzu muß in der Typenidentifikation das durch „KCF_STRING“ definierte Bit gesetzt werden. Ist dies der Fall, so werden die vier Bytes der Code-Tabelle als Zeiger auf eine spezielle Stringtabelle interpretiert. Diese Tabelle besteht aus Einträgen zu je zwei Bytes, wobei das erste Byte die Länge des zugehörigen Strings und das zweite Byte die Startadresse als Offset relativ zum Beginn der Tabelle definiert. Die Anzahl der Einträge hängt von den spezifizierten Qualifiers ab.

```
 0 - Taste alleine gedrückt
 1 - gedrückt mit SHIFT
 2 - gedrückt mit ALT
 3 - gedrückt mit SHIFT und ALT
 4 - gedrückt mit CTRL
 5 - gedrückt mit SHIFT und CTRL
 6 - gedrückt mit ALT und CTRL
 7 - gedrückt mit SHIFT, ALT und CTRL
 8 - Taste alleine losgelassen
 9 - losgelassen mit SHIFT
10 - losgelassen mit ALT
11 - losgelassen mit SHIFT und ALT
12 - losgelassen mit CTRL
13 - losgelassen mit SHIFT und CTRL
14 - losgelassen mit ALT und CTRL
15 - losgelassen mit SHIFT, ALT und CTRL
```

Streicht man aus dieser Tabelle all jene Zeilen, in denen ein nicht spezifizierter Qualifier vorkommt, („losgelassen“ steht hierbei für KCF_DOWNUP), ergibt sich eine Reihenfolge, in der die Einträge zu je zwei Bytes in der Tabelle der Strings auftreten. Die Anzahl der Einträge entspricht also 2 hoch der Anzahl an gesetzten Qualifier-Bits.
Damit läßt sich nun auch der Fall behandeln, wenn mehr als zwei Qualifier in der Typentabelle ohne KCF_STRING gesetzt sind. Dann wird der Eintrag in der Code-Tabelle ebenfalls als Zeiger interpretiert, diesmal jedoch zeigt er auf eine Tabelle, die genau ein Byte je Qualifier-Kombination besitzt, diese jedoch in der gleichen Reihenfolge wie oben beschrieben. Eine Ausnahme bildet hierbei der Fall, daß KCF_SHIFT, KCF_ALT und KCF_CONTROL gesetzt sind, er wird als „KC_VANILLA“ bezeichnet. Hier werden, so als wäre KCF_CONTROL nicht angegeben, lediglich die vier Bytes verwendet und bei gedrückter CTRL-Taste die Bits 5 bis 7 des zugehörigen Codes (in Abhängigkeit der beiden anderen Qualifier) gelöscht, Ausnahmen bestätigen die Regel. Dieser Tastentyp wird für die Buchstabentasten verwendet, da so auf einfache Weise die ASCIIs unter 32 erzeugt werden können.
Auffällig ist, daß diese Tabelle genau der umgekehrten Reihenfolge der vier Bytes im Falle der „einfachen“ Typen (maximal zwei Qualifier) entspricht. Als letzte Besonderheit sind noch die sogenannten „Dead Keys“ zu klären, die erst mit Betriebssystemversion 1.2 eingeführt wurden. Mußte unter Kickstart 1.1 der Anwender noch ein gutes Gedächtnis mitbringen, um sich die Lage all der Sonderzeichen auf der Tastatur zu merken, so bietet die aktuelle Software hierzu ein sehr einfaches Verfahren:
In jeder bisherigen KeyMap dienen fünf Tasten zusammen mit ALT als sogenannte „tote“ Tasten. „Tot“ deshalb, weil sie auf den ersten Blick keine Wirkung zu haben scheinen. Drücken Sie einmal ALT-K und anschließend einen Vokal, er sollte nun mit Trema (Doppelpunkt über dem Buchstaben) erscheinen. Folgende diakritischen Zeichen lassen sich über Dead-Keys erzeugen:

```
ALT-F    Akut
ALT-G    Gravis
ALT-H    Zirkumflex
ALT-J    Tilde
ALT-K    Trema
```

Eine Kombination ist natürlich nicht mit allen Tasten möglich. Sowohl diese Dead-Keys selbst, als auch solche Tasten, auf die das vorherige Drücken von Dead-Keys einen Einfluß hat, werden mit dem Typenbit „KCF_DEAD“ gekennzeichnet. Der Eintrag in der Code-Tabelle ist erwartungsgemäß ein Zeiger auf eine neue Art von Tabelle, eine Dead-Key-Tabelle, die ebenfalls wie die String-Tabelle angeordnet ist und auch zwei Bytes je Eintrag umfaßt. Allerdings wird hier das erste Byte wie folgt interpretiert: Ist Bit 3 gesetzt, so gibt das folgende Byte an, um welches diakritische Zeichen es sich handelt. Dieses Byte wird nochmals in die Nybbles (LSN und MSN) unterteilt interpretiert, was für die Double-Dead-Keys notwendig ist. Ich gehe im Anschluß hierauf nochmals ein. Dieser Wert im LSN liegt im Bereich von eins bis fünf und entspricht den fünf Dead-Keys aus obiger Übersicht in der gleichen Reihenfolge.
Durch Dead-Keys beeinflußte Taste-Qualifier-Kombinationen besitzen ein gesetztes Bit 0 im ersten Byte des Eintrags der Dead-Key-Tabelle. Das zweite Byte zeigt dann auf eine Tabelle mit einem Zeichen mehr als Dead-Key-Typen existieren. Da die Numerierung der Dead-Keys bei 1 beginnt, ist es nicht schwer zu erraten, daß das erste Byte für den nicht-modifizierten (ohne vorhergehenden Dead-Key) Fall gilt. Wer hierhin durchgehalten hat, wird auch mit dem letzten Sonderfall keine Probleme mehr haben: Im Falle der Double-Dead-Keys wird lediglich das LSN des Dead-Typs der ersten Taste mit dem Wert im MSN des zweiten Dead-Modifiers multipliziert und anschließend das LSN dieser zweiten Taste addiert. Dieser Zahlenwert bildet, genau wie das LSN im einfachen Dead-Key-Fall auch, einen Index in besagte Tabelle, die dann jedoch entsprechend länger sein ▶

muß. Als Beispiel soll hier die deutsche Tastatur dienen. Der Routine „RawKeyConvert()“ des console.device ist es übrigens gleichgültig, in welcher Reihenfolge die diakritischen Zeichen angeordnet sind. Die Bedeutung eines Dead-Modifiers offenbart sich erst durch die Tabelle einer modifizierbaren Taste. Somit wäre es durchaus möglich, die Kombination ALT-Komma als weiteren Dead-Key für die Cedille einzuführen. Übrigens gehört auch die Leertaste zu den modifizierbaren Tasten — es erscheint dann das entsprechende diakritische Zeichen alleine. Eine Kombination von KCF_DEAD und KCF_STRING ist aufgrund eines Betriebssystemfehlers zur Zeit nicht möglich.
Als Beispiel soll eine Tastaturbelegung dienen, die sich insbesondere für die Anwender von NanoMon oder anderer sedezimal orientierter Programme als nützlich erwiesen hat: Diese neue Keymap wird wie gewohnt durch „SetMap“ aktiviert und stellt dann auf dem erweiterten Zehnerblock des Amiga 500 und 2000 eine Sedezimaltastatur zur Verfügung. Der abgedruckte Quelltext sollte, im Gegensatz zum Hexdump, auf keinen Fall zum Eintippen verleiten, sondern vielmehr als Referenz dienen. Von der Fehlerträchtigkeit einmal absehend, ist es nicht sicher, daß die vielen Sonderzeichen die Prozedur des Abdrucks selbst mit dem besten Druckertreiber unbeschadet überstehen. Besorgen Sie sich lieber die Leserdiskette, Sie können dann sicher sein, daß die KeyMap auf Anhieb funktioniert.
Der Quelltext ist so angelegt, daß durch Änderung bestimmter Konstanten die KeyMap an deutsche und amerikanische Tastaturen angepaßt werden kann: -durch „KEYMAP_USA1“ und „KEYMAP_D“ kann festgelegt werden, ob es sich um die Belegung für die US-Tastatur oder die DIN-Version handeln soll. -Durch „KEYMAP_HEX“ wird der erweiterte Zehnerblock von Amiga 500 und 2000 bei gleichzeitigem Drücken von SHIFT in eine Sedezimaltastatur umgewandelt. Die Tasten wurden als „CAPSABLE“ (siehe Seite 108) definiert, so daß durch Drücken von „CAPS LOCK“ ebenfalls diese Funktion erreicht werden kann.
Die Keymap „usa1“ der Workbench 1.3 unterscheidet sich in Details von der Version 1.2 und kann durch „KEYMAP_1_3“ erzeugt werden. Bei einer KeyMap handelt es sich, ähnlich einem Font, um ein Executable. Um nun einen Rechnerabsturz beim irrtümlichen Einladen einer KeyMap als CLI-Kommando zu verhindern, kann durch Belegung des Schalters „KEYMAP_CRASH“ mit dem Wert 0 diese derart modifiziert werden, daß sie fortan einen solchen Versuch mit Fehlermeldung zurückweisen wird. Hierzu wird einfach ein kleines Programm in die Knotenstruktur zu Beginn des ersten und einzigen Hunks gesetzt.

Besonders verzwickt ist die Codierung von „Double Dead Keys“ in der deutschen Tastaturbelegung

Die deutsche Tastatur besitzt, im Gegensatz zur US-Version, sogenannte „Double Dead Keys“, Dead Keys also, die durch Drücken einer weiteren „toten Taste“ nochmals modifiziert werden können. Dies ist in diesem Fall die Taste mit den Akzenten „Akut“ und „Gravis“. Auch ist es auf der deutschen Tastatur möglich, durch Drücken von ALT-F und ALT-G einen Zirkumflex zu erzeugen. „KEYMAP_DOUBLE“ ist für die Steuerung dieses Verhaltens zuständig. Letzteres ist nun auch endlich mit der US-Tastatur möglich. Zu beachten ist, daß die deutsche und amerikanische Tastatur natürlich nicht gleichzeitig erzeugt werden können und daß die deutsche Tastatur immer mit aktiviertem KEYMAP_DOUBLE assembliert werden muß.
Als eine weitere Änderung bietet sich die Belegung der Funktionstasten an. Durch Änderung der zugehörigen Key-Types in „KCF_STRING!KCF_SHIFT!KCF_ALT!KCF_CONTROL“ ist es sogar möglich, 80 verschiedene Strings zuzuweisen. Hauptsache, Sie behalten die Übersicht und haben gelenkige Finger!

(Ralph Babel)

Programmname: Keymaps.asm
Sprache: Assembler, Metacomco

Auf Leserdiskette

ABC Eingabe mit beliebigem Editor (ohne Zeilennummern)
Überprüfen mit Checksum (neue Version)

```
 6 * 23-May-1988 created                          000
 7 * 27-May-1988 added KEYMAP_D                   000
 8 *                                              000
 9                                                000
10         nolist                                 613
11                                                613
12         include "exec/types.i"                 073
13         include "exec/execbase.i"              172
14         include "devices/keymap.i"             736
15         include "libraries/dos.i"              317
16         include "libraries/dosextens.i"        237
17                                                237
18         list                                   949
19                                                949
20 *                                              949
21 * constants                                    949
22 *                                              949
23                                                949
24 AbsExecBase      equ     4                     350
25                                                350
26 *                                              350
27 * macros                                       350
28 *                                              350
29                                                350
30 callsys macro                                  839
31         xref    _LVO\1                         249
32         jsr     _LVO\1(a6)                     633
33         endm                                   649
34                                                649
35 *                                              649
36 * switches                                     649
37 *                                              649
38                                                649
39 KEYMAP_USA1      equ     0                     044
40 KEYMAP_D         equ     1                     390
41 KEYMAP_HEX       equ     1
                ;numeric pad (SHIFT/CAPS)         577
42 KEYMAP_1_3       equ     1                     366
43 KEYMAP_CRASH     equ     0                     856
44 KEYMAP_DOUBLE    equ     KEYMAP_D
        ;support double dead keys                 230
45                                                230
46         ifne    KEYMAP_USA1+KEYMAP_D-1         811
47         fail                                   387
48         endc                                   273
49                                                273
50         ifne    KEYMAP_D                       101
51         ifeq    KEYMAP_DOUBLE                  783
52         fail                                   447
53         endc                                   805
54         endc                                   979
55                                                979
56 *                                              979
57 * KeyMapNode structure                         979
58 *                                              979
59                                                979
60         code                                   018
61                                                018
62         ifne    KEYMAP_CRASH                   555
63         dc.l    0
                        ;kn_Node.LN_SUCC          228
64         endc                                   538
65         ifeq    KEYMAP_CRASH                   462
```



```
66          bra.s    1$               ;patch       433
67          dc.w     0                             885
68          endc                                   291
69                                                 291
70          dc.l     0
                          ;kn_Node.LN_PRED         188
71          dc.b     NT_UNKNOWN
                   ;kn_Node.LN_TYPE                160
72          dc.b     0
                          ;kn_Node.LN_PRI          571
73          dc.l     keyMapName
                   ;kn_Node.LN_NAME                124
74                                                 124
75          dc.l     LoKeyMapTypes
              ;kn_KeyMap.km_LoKeyMapTypes          112
76          dc.l     LoKeyMap
                   ;kn_KeyMap.km_LoKeyMap          727
77          dc.l     LoCapsable
                ;kn_KeyMap.km_LoCapsable           550
78          dc.l     LoRepeatable
              ;kn_KeyMap.km_LoRepeatable           791
79          dc.l     HiKeyMapTypes
              ;kn_KeyMap.km_HiKeyMapTypes          053
80          dc.l     HiKeyMap
                   ;kn_KeyMap.km_HiKeyMap          266
81          dc.l     HiCapsable
                ;kn_KeyMap.km_HiCapsable           975
82          dc.l     HiRepeatable
              ;kn_KeyMap.km_HiRepeatable           486
83                                                 486
84 *                                               486
85 * The following piece of code is a patch
   to avoid crashing the                           486
86 * system if the user tries to load a
   keymap as an executable.                        486
87 *                                                486
88                                                 486
89          ifeq     KEYMAP_CRASH                  530
90                                                 530
91 1$       movea.l AbsExecBase,a6                 898
92          suba.l  a1,a1                          979
93          callsys FindTask                       018
94          movea.l d0,a0                          281
95          moveq   #ERROR_FILE_NOT_OBJECT,d0      627
96          move.l  d0,pr_Result2(a0)              960
97          moveq   #-1,d0                         761
98          rts                                    158
99                                                 158
100         endc                                   116
101                                                116
102 *                                              116
103 * 1 = use SHIFT qualifier during CAPS
    LOCK, ignore otherwise                         116
104 *                                              116
105                                                116
106 LoCapsable:                                    036
107         dc.b     %00000000        ;$07-$00:
             7 6 5 4 3 2 1 `                       040
108         dc.b     %00000000        ;$0f-$08:
             0   \ = - 0 9 8                       060
109         dc.b     %11111111        ;$17-$10:
             i u y t r e w q                       536
110         ifne     KEYMAP_USA1                   470
111         dc.b     %00000011        ;$1f-$18:
             3 2 1   ] [ p o                       344
112         endc                                   950
113         ifne     KEYMAP_D                      654
114         dc.b     %00000111        ;$1f-$18:
             3 2 1   ] ü p o                       413
115         endc                                   483
116         dc.b     %11111111        ;$27-$20:
             k j h g f d s a                       627
117         ifne     KEYMAP_USA1                   217
118         dc.b     %00000001        ;$2f-$28:
             6 5 4       ' ; l                     982
119         endc                                   812
120         ifne     KEYMAP_D                      044
121         dc.b     %00000111        ;$2f-$28:
             6 5 4       ä ö l                     411
122         endc                                   705
123         dc.b     %11111110        ;$37-$30:
             m n b v c x z                         676
124         dc.b     %00000000        ;$3f-$38:
             9 8 7 .     / . ,                     192
125                                                192
126 HiCapsable:                                    638
127         dc.b     %00000000        ;$47-$40:    002
128         ifeq     KEYMAP_HEX                    826
129         dc.b     %00000000        ;$4f-$48:
                     -                             942
130         endc                                   108
131         ifne     KEYMAP_HEX                    497
132         dc.b     %00000100        ;$4f-$48:
                     -                             470
133         endc                                   300
134         dc.b     %00000000        ;$57-$50:    360
135         ifeq     KEYMAP_HEX                    568
136         dc.b     %00000000        ;$5f-$58:
             + * / ) (                             652
137         endc                                   578
138         ifne     KEYMAP_HEX                    799
139         dc.b     %01111100        ;$5f-$58:
             + * / ) (                             944
140         endc                                   934
141         dc.b     %00000000        ;$67-$60:    434
142         dc.b     %00000000        ;$6f-$68:    030
143         dc.b     %00000000        ;$77-$70:    010
144                                                010
145 *                                              010
146 * 1 = key is repeatable                        010
147 *                                              010
148                                                010
149 LoRepeatable:                                  844
150         dc.b     %11111111        ;$07-$00:
             7 6 5 4 3 2 1 `                       360
151         dc.b     %10111111        ;$0f-$08:
             0   \ = - 0 9 8                       143
152         dc.b     %11111111        ;$17-$10:
             i u y t r e w q                       975
153         dc.b     %11101111        ;$1f-$18:
             3 2 1   ] [ p o                       434
154         dc.b     %11111111        ;$27-$20:
             k j h g f d s a                       358
155         dc.b     %11101111        ;$2f-$28:
             6 5 4       ' ; l                     421
156         dc.b     %11111111        ;$37-$30:
             m n b v c x z                         317
157         dc.b     %11110111        ;$3f-$38:
             9 8 7 .     / . ,                     908
158                                                908
159 HiRepeatable:                                  708
160         dc.b     %01000111        ;$47-$40:    528
161         dc.b     %11110100        ;$4f-$48:
                     -                             757
162         dc.b     %11111111        ;$57-$50:    949
163         ifeq     KEYMAP_1_3!KEYMAP_D           238
164         dc.b     %00000011        ;$5f-$58:
             + * / ) (                             232
165         endc                                   094
166         ifne     KEYMAP_1_3!KEYMAP_D           408
167         dc.b     %01111111        ;$5f-$58:
             + * / ) (                             195
168         endc                                   105
169         dc.b     %00000000        ;$67-$60:    409
170         dc.b     %00000000        ;$6f-$68:    881
171         dc.b     %00000000        ;$77-$70:    193
172                                                193
173 *                                              193
174 * key types                                    193
175 *                                              193
176 * KC_NOQUAL     EQU     0       ;not to
    be affected by qualifiers                      193
177 * DC_VANILLA    EQU     7       ;see
    below: CTRL+key clears bits 5 to 7             193
178 *                                              193
179 * KCF_SHIFT     EQU     $01     ;SHIFT
    generates separate code                        193
180 * KCF_ALT       EQU     $02     ;ALT
    generates separate code                        193
181 * KCF_CONTROL   EQU     $04     ;CTRL
    generates separate code                        193
182 * KCF_DOWNUP    EQU     $08
         ;releasing the key generates
    separate code                                  193
183 *                                              193
184 * KCF_DEAD      EQU     $20     ;may be
    dead or modified by dead keys                  193
185 * KCF_STRING    EQU     $40     ;key is
    mapped to string instead of char               193
186 * KCF_NOP       EQU     $80     ;don't
    map this key!                                  193
187 *                                              193
188                                                193
189 LoKeyMapTypes:                                 810
190         dc.b     KC_VANILLA
                       ;$00: ` ~                   077
191         dc.b     KCF_SHIFT!KCF_ALT
                ;$01: 1 !                          214
192         ifne     KEYMAP_USA1                   020
193         dc.b     KC_VANILLA
                       ;$02: 2 @                   927
194         endc                                   773
195         ifne     KEYMAP_D                      649
196         dc.b     KCF_SHIFT!KCF_ALT
                ;$02: 2 "                           386
197         endc                                   384
198         dc.b     KCF_SHIFT!KCF_ALT
                ;$03: 3 #                          693
199         dc.b     KCF_SHIFT!KCF_ALT
                ;$04: 4 $                          502
200         dc.b     KCF_SHIFT!KCF_ALT
                ;$05: 5 %                          731
201         dc.b     KC_VANILLA
                       ;$06: 6 ^                   062
202         dc.b     KCF_SHIFT!KCF_ALT
                ;$07: 7 &                          811
203                                                811
204         dc.b     KCF_SHIFT!KCF_ALT
                ;$08: 8 *                          492
205         dc.b     KCF_SHIFT!KCF_ALT
                ;$09: 9 (                          081
206         dc.b     KCF_SHIFT!KCF_ALT
                ;$0a: 0 )                          874
207         dc.b     KC_VANILLA
                       ;$0b: - _                   141
208         ifne     KEYMAP_USA1                   395
209         dc.b     KCF_SHIFT
                       ;$0c: = +                   807
210         endc                                   757
211         ifne     KEYMAP_D                      697
212         dc.b
             KCF_SHIFT!KCF_ALT!KCF_DEAD
                ;$0c: ' `                           532
213         endc                                   562
214         dc.b     KC_VANILLA
                       ;$0d: \ |                   341
215         dc.b     KCF_NOP
                       ;$0e:
            (undefined)                            536
216         dc.b     KC_NOQUAL
                       ;$0f: 0 (numeric
             pad)                                  640
217                                                640
218         dc.b     KC_VANILLA
                       ;$10: q Q                   267
219         dc.b     KC_VANILLA
                       ;$11: w W                   622
220         dc.b     KC_VANILLA!KCF_DEAD
                ;$12: e E                          463
221         dc.b     KC_VANILLA
                       ;$13: r R                   026
222         dc.b     KC_VANILLA
                       ;$14: t T                   317
223         ifne     KEYMAP_USA1                   547
224         dc.b     KC_VANILLA!KCF_DEAD
                ;$15: y Y                          120
225         endc                                   238
226         ifne     KEYMAP_D                      742
227         dc.b     KC_VANILLA
                       ;$15: z Z                   753
228         endc                                   607
229         dc.b     KC_VANILLA!KCF_DEAD
                ;$16: u U                          004
230         dc.b     KC_VANILLA!KCF_DEAD
                ;$17: i I                          453
231                                                453
232         dc.b     KC_VANILLA!KCF_DEAD
                ;$18: o O                          370
233         dc.b     KC_VANILLA
                       ;$19: p P                   149
234         dc.b     KC_VANILLA
                       ;$1a: [ {                   096
235         dc.b     KC_VANILLA
                       ;$1b: ] }                   771
236         dc.b     KCF_NOP
                       ;$1c:
            (undefined)                            270
237         dc.b     KC_NOQUAL
                       ;$1d: 1 (numeric
             pad)                                  310
238         dc.b     KC_NOQUAL
                       ;$1e: 2 (numeric
             pad)                                  510
239         dc.b     KC_NOQUAL
                       ;$1f: 3 (numeric
             pad)                                  510
240                                                510
241         dc.b     KC_VANILLA!KCF_DEAD
                ;$20: a A                          543
242         dc.b     KC_VANILLA
                       ;$21: s S                   130
243         dc.b     KC_VANILLA
                       ;$22: d D                   933
244         dc.b     KC_VANILLA!KCF_DEAD
                ;$23: f F                          442
245         dc.b     KC_VANILLA!KCF_DEAD
                ;$24: g G                          811
246         dc.b     KC_VANILLA!KCF_DEAD
                ;$25: h H                          200
247         dc.b     KC_VANILLA!KCF_DEAD
                ;$26: j J                          801
248         dc.b     KC_VANILLA!KCF_DEAD
                ;$27: k K                          926
249                                                926
250         dc.b     KC_VANILLA
                       ;$28: l L                   425
251         ifne     KEYMAP_USA1                   423
252         dc.b     KCF_SHIFT
                       ;$29: ; :                   947
253         dc.b     KCF_SHIFT
                       ;$2a: ' "                   519
254         dc.b     KCF_NOP
                       ;$2b: (reserved)            026
255         endc                                   432
256         ifne     KEYMAP_D                      136
257         dc.b     KCF_SHIFT!KCF_ALT
                ;$29: ö Ö                          733
258         dc.b     KCF_SHIFT!KCF_ALT
                ;$2a: ä Ä                          958
259         dc.b     KC_VANILLA
                       ;$2b: # ^                   457
260         endc                                   695
261         dc.b     KCF_NOP
                       ;$2c:
            (undefined)                            498
262         dc.b     KC_NOQUAL
                       ;$2d: 4 (numeric
             pad)                                  450
263         dc.b     KC_NOQUAL
                       ;$2e: 5 (numeric
             pad)                                  210
264         dc.b     KC_NOQUAL
                       ;$2f: 6 (numeric
             pad)                                  010
265                                                010
266         ifne     KEYMAP_USA1                   048
267         dc.b     KCF_SHIFT!KCF_ALT
                ;$30: (reserved)                   549
268         dc.b     KC_VANILLA
                       ;$31: z Z                   520
269         endc                                   894
270         ifne     KEYMAP_D                      822
271         dc.b     KCF_SHIFT
                       ;$30: < >                   918
272         dc.b     KC_VANILLA!KCF_DEAD
                ;$31: y Y                          983
273         endc                                   701
274         dc.b     KC_VANILLA
                       ;$32: x X                   160
275         dc.b     KC_VANILLA
                       ;$33: c C                   835
276         dc.b     KC_VANILLA
                       ;$34: v V                   678
277         dc.b     KC_VANILLA
                       ;$35: b B                   689
278         dc.b     KC_VANILLA!KCF_DEAD
                ;$36: n N                          006
279         dc.b     KC_VANILLA
                       ;$37: m M                   017
280                                                017
281         ifne     KEYMAP_USA1                   527
282         dc.b     KCF_SHIFT
                       ;$38: , <                   467
283         dc.b     KCF_SHIFT
                       ;$39: . >                   167
284         dc.b     KCF_SHIFT
                       ;$3a: / ?                   715
285         endc                                   729
286         ifne     KEYMAP_D                      781
287         dc.b     KCF_SHIFT!KCF_ALT
                ;$38: , <                          686
288         dc.b     KCF_SHIFT!KCF_ALT
                ;$39: . >                          371
289         dc.b     KC_VANILLA
                       ;$3a: / ?                   190
290         endc                                   660
291         dc.b     KCF_NOP
                       ;$3b:
            (undefined)                            091
292         dc.b     KC_NOQUAL
                       ;$3c: . (numeric
             pad)                                  439
293         dc.b     KC_NOQUAL
                       ;$3d: 7 (numeric
             pad)                                  131
294         dc.b     KC_NOQUAL
                       ;$3e: 8 (numeric
             pad)                                  639
295         dc.b     KC_NOQUAL
                       ;$3f: 9 (numeric
             pad)                                  131
296                                                131
297 HiKeyMapTypes:                                 666
298         dc.b     KCF_ALT!KCF_DEAD
                ;$40: Space Bar                    231
299         dc.b     KC_NOQUAL
                       ;$41: BACKSPACE             139
300         dc.b     KCF_SHIFT!KCF_STRING
              ;$42: TAB                            666
301         dc.b     KC_NOQUAL
                       ;$43: ENTER                 242
302         dc.b     KCF_CONTROL
                       ;$44: RETURN                017
303         dc.b     KCF_ALT
                       ;$45: ESC                   164
304         dc.b     KC_NOQUAL
                       ;$46: DEL                   828
```



```
305       dc.b     KCF_NOP
                            ;$47:
   (undefined)                                   635
306                                              635
307       dc.b     KCF_NOP
                            ;$48:
   (undefined)                                   678
308       dc.b     KCF_NOP
                            ;$49:
   (undefined)                                   061
309       ifeq     KEYMAP_HEX                    097
310       dc.b     KC_NOQUAL
                            ;$4a: - (numeric
   pad)                                          469
311       endc                                   411
312       ifne     KEYMAP_HEX                    988
313       dc.b     KCF_SHIFT
                            ;$4a: - (numeric
   pad)                                          700
314       endc                                   346
315       dc.b     KCF_NOP
                            ;$4b:
   (undefined)                                   009
316       dc.b     KCF_SHIFT!KCF_STRING
              ;$4c: Up Arrow                     544
317       dc.b     KCF_SHIFT!KCF_STRING
              ;$4d: Down Arrow                   239
318       dc.b     KCF_SHIFT!KCF_STRING
              ;$4e: Forward Arrow                566
319       dc.b     KCF_SHIFT!KCF_STRING
              ;$4f: Backward Arrow               437
320                                              437
321       dc.b     KCF_SHIFT!KCF_STRING
              ;$50: F1                           300
322       dc.b     KCF_SHIFT!KCF_STRING
              ;$51: F2                           091
323       dc.b     KCF_SHIFT!KCF_STRING
              ;$52: F3                           234
324       dc.b     KCF_SHIFT!KCF_STRING
              ;$53: F4                           497
325       dc.b     KCF_SHIFT!KCF_STRING
              ;$54: F5                           840
326       dc.b     KCF_SHIFT!KCF_STRING
              ;$55: F6                           855
327       dc.b     KCF_SHIFT!KCF_STRING
              ;$56: F7                           966
328       dc.b     KCF_SHIFT!KCF_STRING
              ;$57: F8                           941
329                                              941
330       dc.b     KCF_SHIFT!KCF_STRING
              ;$58: F9                           740
331       dc.b     KCF_SHIFT!KCF_STRING
              ;$59: F10                          955
332       ifne     KEYMAP_USA1                   057
333       dc.b     KCF_SHIFT!KCF_ALT
              ;$5a: ( (numeric pad)              010
334       dc.b     KCF_SHIFT!KCF_ALT
              ;$5b: ) (numeric pad)              423
335       dc.b     KCF_SHIFT!KCF_ALT
              ;$5c: / (numeric pad)              008
336       dc.b     KCF_SHIFT!KCF_ALT
              ;$5d: * (numeric pad)              093
337       dc.b     KCF_SHIFT!KCF_ALT
              ;$5e: + (numeric pad)              806
338       endc                                   868
339       ifeq     KEYMAP_USA1                   113
340       ifne     KEYMAP_D                      581
341       ifeq     KEYMAP_HEX                    649
342       dc.b     KCF_SHIFT!KCF_CONTROL
             ;$5a: [ [ (numeric pad)             722
343       dc.b     KCF_SHIFT!KCF_CONTROL
             ;$5b: ] ] (numeric pad)             047
344       dc.b     KC_NOQUAL
                            ;$5c: / (numeric
   pad)                                          219
345       dc.b     KC_NOQUAL
                            ;$5d: * (numeric
   pad)                                          079
346       dc.b     KC_NOQUAL
                            ;$5e: + (numeric
   pad)                                          379
347       endc                                   161
348       ifne     KEYMAP_HEX                    786
349       dc.b     KC_VANILLA
                            ;$5a: [ { (numeric
   pad)                                          565
350       dc.b     KC_VANILLA
                            ;$5b: ] } (numeric
   pad)                                          512
351       dc.b     KCF_SHIFT
                            ;$5c: / (numeric
   pad)                                          368
352       dc.b     KCF_SHIFT
                            ;$5d: * (numeric
   pad)                                          696
353       dc.b     KCF_SHIFT
                            ;$5e: + (numeric
   pad)                                          288
354       endc                                   022
355       endc                                   876
356       endc                                   290
357       dc.b     KCF_STRING
                            ;$5f: HELP           683
358                                              683
359       dc.b     KCF_NOP
                            ;$60: Left SHIFT     534
360       dc.b     KCF_NOP
                            ;$61: Right
   SHIFT                                         493
361       dc.b     KCF_NOP
                            ;$62: CAPS LOCK      080
362       dc.b     KCF_NOP
                            ;$63: CTRL           831
363       dc.b     KCF_NOP
                            ;$64: Left ALT       090
364       dc.b     KCF_NOP
                            ;$65: Right ALT      777
365       dc.b     KCF_NOP
                            ;$66: Left AMIGA
         (C=)                                    276
366           dc.b     KCF_NOP
                            ;$67: Right
         AMIGA                                   243
367                                              243
368           dc.b     KCF_NOP
                            ;$68:
      (undefined)                                854
369           dc.b     KCF_NOP
                            ;$69:
      (undefined)                                533
370           dc.b     KCF_NOP
                            ;$6a:
      (undefined)                                008
371           dc.b     KCF_NOP
                            ;$6b:
      (undefined)                                047
372           dc.b     KCF_NOP
                            ;$6c:
      (undefined)                                394
373           dc.b     KCF_NOP
                            ;$6d:
      (undefined)                                913
374           dc.b     KCF_NOP
                            ;$6e:
      (undefined)                                868
375           dc.b     KCF_NOP
                            ;$6f:
      (undefined)                                515
376                                              515
377           dc.b     KCF_NOP
                            ;$70:
      (undefined)                                038
378           dc.b     KCF_NOP
                            ;$71:
      (undefined)                                933
379           dc.b     KCF_NOP
                            ;$72:
      (undefined)                                808
380           dc.b     KCF_NOP
                            ;$73:
      (undefined)                                695
381           dc.b     KCF_NOP
                            ;$74:
      (undefined)                                498
382           dc.b     KCF_NOP
                            ;$75:
      (undefined)                                601
383           dc.b     KCF_NOP
                            ;$76:
      (undefined)                                804
384           dc.b     KCF_NOP
                            ;$77:
      (undefined)                                707
385                                              707
386 *                                            707
387 * The actual key map                         707
388 *                                            707
389 * If the KeyMapType is either KCF_DEAD or
    KCF_STRING,                                  707
390 * the entry is a pointer, otherwise 4
    bytes of data.                               707
391 *                                            707
392 * table of possible qualifier
    combinations                                 707
393 *                                            707
394 * KC_NOQUAL                 -           -
      -        alone                             707
395 * KCF_SHIFT                 -           -
      Shift  alone                               707
396 * KCF_ALT                   -           -
      Alt    alone                               707
397 * KCF_CONTROL               -           -
      Ctrl   alone                               707
398 * KCF_DOWNUP                -           -
      Up     alone                               707
399 * KCF_SHIFT ¦ KCF_ALT       Shift+Alt
    Alt   Shift  alone                           707
400 * KCF_SHIFT ¦ KCF_CONTROL   Shift+Ctrl
    Ctrl  Shift  alone                           707
401 * KCF_SHIFT ¦ KCF_DOWNUP    Shift+Up
     Up    Shift  alone                          707
402 * KCF_ALT ¦ KCF_CONTROL     Alt+Ctrl
     Ctrl  Alt    alone                          707
403 * KCF_ALT ¦ KCF_DOWNUP      Alt+Up
      Up    Alt    alone                         707
404 * KCF_CONTROL ¦ KCF_DOWNUP  Ctrl+Up
      Up    Ctrl   alone                         707
405 * KC_VANILLA                Shift+Alt
    Alt   Shift  alone                           707
406 *                                            707
407 * Specifying more than two qualifier bits
    in the table of                              707
408 * KeyMapTypes (except for KC_VANILLA)
    will cause the                               707
409 * four-byte-entry being interpreted as a
    pointer to an array                          707
410 * of chars where each byte (same order as
    KCF_STRING and                               707
411 * KCF_DEAD) corresponds to the
    appropriate qualifier                        707
412 * combination.                               707
413 *                                            707
414                                              707
415           cnop     0,2                       329
416                                              329
417 LoKeyMap:                                    797
418           dc.b     '~','`','~','`'
                ;$00: ` ~                        044
419           dc.b     '!','1','!','1'
                ;$01: 1 !                        047
420           ifne     KEYMAP_USA1               429
421           dc.b     '@','/','@','2'
                ;$02: 2 @                        780
422           dc.b     '#','3','#','3'
                ;$03: 3 #                        799
423           dc.b     '$',',','$','4'
                ;$04: 4 $                        168
424           endc                               006
425           ifne     KEYMAP_D                  390
426           dc.b     '/','@','"','2'
                ;$02: 2 "                        379
427           dc.b     '#','3','§','3'
                ;$03: 3 §                        942
428           dc.b     ','x','$','4'
                ;$04: 4 $                        319
429           endc                               677
430           dc.b     '%',',','%','5'
                ;$05: 5 %                        915
431           ifne     KEYMAP_USA1               505
432           dc.b     '^','+','^','6'
                ;$06: 6 ^                         267
433          dc.b     '&',' ','&','7'
                ;$07: 7 &                        895
434          dc.b     '*','y','*','8'
                ;$08: 8 *                        789
435          dc.b     '(','.','(','9'
                ;$09: 9 (                        556
436          dc.b     ')','/',')','0'
                ;$0a: 0 )                        376
437          endc                                494
438          ifne     KEYMAP_D                   974
439          dc.b     '^','+','&','6'
                ;$06: 6 &                        308
440          dc.b     '&',' ','/','7'
                ;$07: 7 /                        573
441          dc.b     '*','y','(','8'
                ;$08: 8 (                        577
442          dc.b     '(','.',')','9'
                ;$09: 9 )                        737
443          dc.b     ')','/','=','0'
                ;$0a: 0 =                        029
444          endc                                099
445                                              099
446          ifne     KEYMAP_USA1                713
447          dc.b     '_','-','_','-'
                ;$0b: - _                        532
448          dc.b     '+','=','+','='
                ;$0c: = +                        907
449          endc                                945
450          ifne     KEYMAP_D                   133
451          dc.b     '_','-','?','ß'
                ;$0b: - _                        194
452          dc.l     keydata_accent
                 ;$0c: ' `                       651
453          endc                                689
454          dc.b     '¦','\','¦','\'
                ;$0d: \ ¦                        748
455          dc.b     0,0,0,0
                            ;$0e:
     (undefined)                                 275
456          dc.b     0,0,0,'0'
                            ;$0f: 0 (numeric
     pad)                                        996
457                                              996
458          dc.b     'Å','å','Q','q'
                ;$10: q Q                        599
459          dc.b     'x','x','W','w'
                ;$11: w W                        216
460          dc.l     keydata_e
                            ;$12: e E            488
461          dc.b     'R','R','R','r'
                ;$13: r R                        383
462          dc.b     'Þ','þ','T','t'
                ;$14: t T                        606
463          ifne     KEYMAP_USA1                924
464          dc.l     keydata_y
                            ;$15: y Y            192
465          endc                                438
466          ifne     KEYMAP_D                   142
467          dc.b     '¥','¤','Z','z'
                ;$31: z Z                        018
468          endc                                272
469          dc.l     keydata_u
                            ;$16: u U            976
470          dc.l     keydata_i
                            ;$17: i I            244
471                                              244
472          dc.l     keydata_o
                            ;$18: o O            758
473          dc.b     'c','c','P','p'
                ;$19: p P                        865
474          ifne     KEYMAP_USA1                351
475          dc.b     '{','[','{','['
                ;$1a: [ {                        646
476          dc.b     '}',']','}',']'
                ;$1b: ] }                        121
477          endc                                127
478          ifne     KEYMAP_D                   563
479          dc.b     '{','[','Ü','ü'
                ;$1a: ü Ü                        772
480          dc.b     '}',']','*','+'
                ;$1b: + *                        218
481          endc                                600
482          dc.b     0,0,0,0
                            ;$1c:
     (undefined)                                 719
483          dc.b     0,0,0,'1'
                            ;$1d: 1 (numeric
     pad)                                        337
484          dc.b     0,0,0,'2'
                            ;$1e: 2 (numeric
     pad)                                        644
485          dc.b     0,0,0,'3'
                            ;$1f: 3 (numeric
     pad)                                        276
486                                              276
487          dc.l     keydata_a
                            ;$20: a A            736
488          dc.b     '§','ß','S','s'
                ;$21: s S                        899
489          dc.b     'Ð','k','D','d'
                ;$22: d D                        470
490          dc.l     keydata_f
                            ;$23: f F            723
491          dc.l     keydata_g
                            ;$24: g G            977
492          dc.l     keydata_h
                            ;$25: h H            212
493          dc.l     keydata_j
                            ;$26: j J            533
494          dc.l     keydata_k
                            ;$27: k K            079
495                                              079
496          dc.b     '£','£','L','l'
                ;$28: l L                        384
497          ifne     KEYMAP_USA1                830
498          dc.b     ':',';',':',';'
                ;$29: ; :                        015
499          dc.b     '"','''','"',''''
               ;$2a: ' "                         634
500          dc.b     0,0,0,0
                            ;$2b: (reserved)     593
501          endc                                983
502          ifne     KEYMAP_D                   043
503          dc.b     ':',';','Ö','ö'
                ;$29: ö Ö                        663
504          dc.b     '"','''','Ä','ä'
               ;$2a: ä Ä                         882
505          dc.b     '^','#','^','#'
                ;$2b: # ^                        435
506          endc                                089
507          dc.b     0,0,0,0
                            ;$2c:
     (undefined)                                 692
```

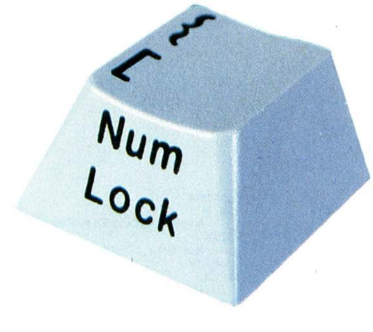
```
508         dc.b    0,0,0,'4'
                         ;$2d: 4 (numeric
             pad)                              685
509         dc.b    0,0,0,'5'
                         ;$2e: 5 (numeric
             pad)                              843
510         dc.b    0,0,0,'6'
                         ;$2f: 6 (numeric
             pad)                              754
511                                            754
512         ifne    KEYMAP_USA1                304
513         dc.b    '/','.','>','<'
                     ;$30: (reserved)          311
514         dc.b    '*','q','Z'.'z'
                     ;$31: z Z                 171
515         endc                               721
516         ifne    KEYMAP_D                   805
517         dc.b    0,0,'>','<'
                       ;$30: < >               950
518         dc.l    keydata_y
                         ;$31: z Z             322
519         endc                               344
520         dc.b    'v','X','X','x'
                     ;$32: x X                 421
521         dc.b    '','ç','C','c'
                     ;$33: c C                 692
522         dc.b    '&','&','V','v'
                     ;$34: v V                 411
523         dc.b    '','','B','b'
                     ;$35: b B                 726
524         dc.l    keydata_n
                         ;$36: n N             059
525         dc.b    '(',',','M','m'
                     ;$37: m M                 447
526                                            447
527         ifne    KEYMAP_USA1                805
528         dc.b    '<',',','<',','
                     ;$38: , <                 024
529         dc.b    '>','.','>','.'
                     ;$39: . >                 575
530         dc.b    '?','/','?','/'
                     ;$3a: / ?                 558
531         endc                               772
532         ifne    KEYMAP_D                   164
533         dc.b    '<',',',';',','
                     ;$38: , ;                 834
534         dc.b    '>','.',':','.'
                     ;$39: . :                 541
535         dc.b    '?','/','_','-'
                     ;$3a: - _                 290
536         endc                               800
537         dc.b    0,0,0,0
                           ;$3b:
             (undefined)                       119
538         dc.b    0,0,0,'.'
                         ;$3c: . (numeric
             pad)                              926
539         dc.b    0,0,0,'7'
                         ;$3d: 7 (numeric
             pad)                              314
540         dc.b    0,0,0,'8'
                         ;$3e: 8 (numeric
             pad)                              519
541         dc.b    0,0,0,'9'
                         ;$3f: 9 (numeric
             pad)                              689
542                                            689
543 HiKeyMap:                                  387
544         dc.l    keydata_space
                       ;$40: Space Bar         398
545         dc.b    0,0,0,$08
                         ;$41: BACKSPACE       361
546         dc.l    keydata_tab
                       ;$42: TAB               427
547         dc.b    0,0,0,$0d
                         ;$43: ENTER           054
548         dc.b    0,0,$0a,$0d
                      ;$44: RETURN             798
549         dc.b    0,0,$9b,$1b
                      ;$45: ESC                595
550         dc.b    0,0,0,$7f
                        ;$46: DEL              055
551         dc.b    0,0,0,0
                          ;$47:
             (undefined)                       818
552                                            818
553         dc.b    0,0,0,0
                          ;$48:
             (undefined)                       041
554         dc.b    0,0,0,0
                          ;$49:
             (undefined)                       836
555         ifne    KEYMAP_HEX                 041
556         dc.b    0,0,'e','-'
                      ;$4a: - (numeric
             pad)                              586
557         endc                               712
558         ifeq    KEYMAP_HEX                 280
559         dc.b    0,0,0,'-'
                        ;$4a: - (numeric
             pad)                              538
560         endc                               944
561         dc.b    0,0,0,0
                          ;$4b:
             (undefined)                       687
562         dc.l    keydata_up
                        ;$4c: Up Arrow         511
563         dc.l    keydata_down
                      ;$4d: Down Arrow         830
564         dc.l    keydata_right
                     ;$4e: Forward Arrow       375
565         dc.l    keydata_left
                      ;$4f: Backward Arrow     833
566                                            833
567         dc.l    keydata_f1
                        ;$50: F1               503
568         dc.l    keydata_f2
                        ;$51: F2               546
569         dc.l    keydata_f3
                        ;$52: F3               282
570         dc.l    keydata_f4
                        ;$53: F4               567
571         dc.l    keydata_f5
                        ;$54: F5               649
572         dc.l    keydata_f6
                        ;$55: F6               896
573         dc.l    keydata_f7
                        ;$56: F7               556
574         dc.l    keydata_f8
                        ;$57: F8               997
575                                            997
576         dc.l    keydata_f9
                        ;$58: F9               027
577         dc.l    keydata_f10
                      ;$59: F10                213
578         ifne    KEYMAP_HEX                 046
579         ifne    KEYMAP_USA1                340
580         dc.b    '(','(','a','('
                    ;$5a: ( (numeric pad)      028
581         dc.b    ')',')','b',')'
                    ;$5b: ) (numeric pad)      920
582         dc.b    '/','/','c','/'
                    ;$5c: / (numeric pad)      799
583         dc.b    '*','*','d','*'
                    ;$5d: * (numeric pad)      324
584         dc.b    '+','+','f','+'
                    ;$5e: + (numeric pad)      178
585         endc                               944
586         ifne    KEYMAP_D                   648
587         dc.b    '{','{','a','['
                    ;$5a: [ { (numeric pad)    001
588         dc.b    '}','}','b',']'
                    ;$5b: ] } (numeric pad)    925
589         dc.b    0,0,'c','/'
                      ;$5c: / (numeric
             pad)                              038
590         dc.b    0,0,'d','*'
                      ;$5d: * (numeric
             pad)                              943
591         dc.b    0,0,'f','+'
                      ;$5e: + (numeric
             pad)                              255
592         endc                               589
593         endc                               379
594         ifeq    KEYMAP_HEX                 639
595         ifne    KEYMAP_USA1                997
596         dc.b    '(','(','(','('
                    ;$5a: ( (numeric pad)      168
597         dc.b    ')',')',')',')'
                    ;$5b: ) (numeric pad)      603
598         dc.b    '/','/','/','/'
                    ;$5c: / (numeric pad)      546
599         dc.b    '*','*','*','*'
                    ;$5d: * (numeric pad)      793
600         dc.b    '+','+','+','+'
                    ;$5e: + (numeric pad)      152
601         endc                               782
602         ifne    KEYMAP_D                   110
603         dc.b    $1b,$1b,'{','['
                    ;$5a: [ { (numeric pad)    157
604         dc.b    $1d,$1d,'}',']'
                    ;$5b: ] } (numeric pad)    624
605         dc.b    0,0,0,'/'
                        ;$5c: / (numeric
             pad)                              548
606         dc.b    0,0,0,'*'
                        ;$5d: * (numeric
             pad)                              323
607         dc.b    0,0,0,'+'
                        ;$5e: + (numeric
             pad)                              391
608         endc                               877
609         endc                               179
610         dc.l    keydata_help
                      ;$5f: HELP               627
611                                            627
612         dc.b    0,0,0,0
                          ;$60: Left SHIFT     102
613         dc.b    0,0,0,0
                          ;$61: Right
             SHIFT                             141
614         dc.b    0,0,0,0
                          ;$62: CAPS LOCK      536
615         dc.b    0,0,0,0
                          ;$63: CTRL           911
616         dc.b    0,0,0,0
                          ;$64: Left ALT       298
617         dc.b    0,0,0,0
                          ;$65: Right ALT      601
618         dc.b    0,0,0,0
                          ;$66: Left AMIGA
             (C=)                              204
619         dc.b    0,0,0,0
                          ;$67: Right
             AMIGA                             859
620                                            859
621         dc.b    0,0,0,0
                          ;$68:
             (undefined)                       454
622         dc.b    0,0,0,0
                          ;$69:
             (undefined)                       133
623         dc.b    0,0,0,0
                          ;$6a:
             (undefined)                       560
624         dc.b    0,0,0,0
                          ;$6b:
             (undefined)                       839
```

```
625         dc.b    0,0,0,0
                            ;$6c:
        (undefined)                             514
626         dc.b    0,0,0,0
                            ;$6d:
        (undefined)                             953
627         dc.b    0,0,0,0
                            ;$6e:
        (undefined)                             148
628         dc.b    0,0,0,0
                            ;$6f:
        (undefined)                             027
629                                             027
630         dc.b    0,0,0,0
                            ;$70:
        (undefined)                             902
631         dc.b    0,0,0,0
                            ;$71:
        (undefined)                             837
632         dc.b    0,0,0,0
                            ;$72:
        (undefined)                             496
633         dc.b    0,0,0,0
                            ;$73:
        (undefined)                             031
634         dc.b    0,0,0,0
                            ;$74:
        (undefined)                             890
635         dc.b    0,0,0,0
                            ;$75:
        (undefined)                             873
636         dc.b    0,0,0,0
                            ;$76:
        (undefined)                             388
637         dc.b    0,0,0,0
                            ;$77:
        (undefined)                             307
638                                             307
639 *                                           307
640 * Table of Dead Key Data                    307
641 *                                           307
642 * The console.device's "RawKeyConvert()"
    does NOT IMPOSE                             307
643 * ANY MEANING to the byte following
    "DPF_DEAD", it depends                      307
644 * on the table specified by the offset
    after "DPF_MOD"!                            307
645 *                                           307
646 * Dead Prefix Bytes                         307
647 *                                           307
648 * DPF_MOD       EQU     $01     ;key is
    modified by dead key                        307
649 * DPF_DEAD      EQU     $08     ;key is
    dead                                        307
650 *                                           307
651 * DP_2DINDEXMASK  EQU   $0F     ;mask for
    index for 1st of two dead keys              307
652 * DP_2DFACSHIFT   EQU   4       ;shift
    for factor for 1st of two dead keys         307
653 *                                           307
654 * If KCF_STRING is specified in the table
    of KeyTypes, the                            307
655 * byte in the table pointed to by the
    offset following the                        307
656 * Dead Prefix "DPF_MOD" points to a
    string descriptor. The                      307
657 * base address for the string offset is
    the address of the                          307
658 * string descriptor itself! Dead Prefix
    "0" means: Use                              307
659 * second byte as char - or offset to a
    two-byte string                             307
660 * descriptor. This time the base address
    for the offset is                           307
661 * the start address of the Dead Table!
    The combination of                          307
662 * KCF_DEAD and KCF_STRING does not work
    under Kickstart 1.2                         307
663 * due to a bug in the console.device's
    "RawKeyConvert()"                           307
664 * (locations $fe7356 and $fe73b4 should
    read "BTST                                  307
665 * #KCB_STRING,D2"). - Too bad!              307
666 *                                           307
667                                             307
668 DEAD_ACUTE      equ     1                   158
669 DEAD_GRAVE      equ     2                   753
670 DEAD_CIRCUMFLEX equ     3                   678
671 DEAD_TILDE      equ     4                   852
672 DEAD_TREMA      equ     5       ;better
    known as "umlaut"                           458
673 DEAD_MAX        equ     6       ;for
    "double dead"                               646
674                                             646
675 *                                           646
676 * order of table entries (delete
    combinations with qualifiers not
    specified):                                 646
677 *                                           646
678 *  0 - alone             8 - Up             646
679 *  1 - Shift             9 - Shift+Up       646
680 *  2 - Alt              10 - Alt+Up         646
681 *  3 - Shift+Alt        11 -
    Shift+Alt+Up                                646
682 *  4 - Control          12 - Control+Up     646
683 *  5 - Shift+Control    13 -
    Shift+Control+Up                            646
684 *  6 - Alt+Control      14 -
    Alt+Control+Up                              646
685 *  7 - Shift+Alt+Control  15
    Shift+Alt+Control+Up                        646
686 *                                           646
687                                             646
688 *                                           646
689 * dead keys                                 646
690 *                                           646
691                                             646
692         ifne    KEYMAP_D                    518
693 keydata_accent:                             304
694         dc.b
                DPF_DEAD,DEAD_MAX<<DP_2DFACSH
        IFT!DEAD_ACUTE                          710
695         dc.b
                DPF_DEAD,DEAD_MAX<<DP_2DFACSH
        IFT!DEAD_GRAVE                          507
696         dc.b    0,'=',0,'+'                 002
697         endc                                000
698                                             000
699 keydata_f:                                  754
700         dc.b    0,'f',0,'F'                 837
701         ifeq    KEYMAP_DOUBLE               239
702         dc.b
                DPF_DEAD,DEAD_ACUTE,DPF_DEAD,
        DEAD_ACUTE                              710
703         endc                                284
704         ifne    KEYMAP_DOUBLE               415
705         dc.b
                DPF_DEAD,DEAD_MAX<<DP_2DFACSH
        IFT!DEAD_ACUTE                          629
706         dc.b
                DPF_DEAD,DEAD_MAX<<DP_2DFACSH
        IFT!DEAD_ACUTE                          971
707         endc                                985
708         dc.b    0,$06,0,$06,0,$86,0,$86     876
709                                             876
710 keydata_g:                                  691
711         dc.b    0,'g',0,'G'                 720
712         ifeq    KEYMAP_DOUBLE               114
713         dc.b
                DPF_DEAD,DEAD_GRAVE,DPF_DEAD,
        DEAD_GRAVE                              407
714         endc                                101
715         ifne    KEYMAP_DOUBLE               272
716         dc.b
                DPF_DEAD,DEAD_MAX<<DP_2DFACSH
        IFT!DEAD_GRAVE                          941
717         dc.b
                DPF_DEAD,DEAD_MAX<<DP_2DFACSH
        IFT!DEAD_GRAVE                          010
718         endc                                160
719         dc.b    0,$07,0,$07,0,$87,0,$87     279
720                                             279
721 keydata_h:                                  283
722         dc.b
                0,'h',0,'H',DPF_DEAD,DEAD_CIR
        CUMFLEX,DPF_DEAD,DEAD_CIRCUMFLEX        982
723         dc.b    0,$08,0,$08,0,$88,0,$88     089
724                                             089
725 keydata_j:                                  495
726         dc.b
                0,'j',0,'J',DPF_DEAD,DEAD_TIL
        DE,DPF_DEAD,DEAD_TILDE                  398
727         dc.b    0,$0A,0,$0A,0,$8A,0,$8A     109
728                                             109
729 keydata_k:                                  456
730         dc.b
                0,'k',0,'K',DPF_DEAD,DEAD_TRE
        MA,DPF_DEAD,DEAD_TREMA                  327
731         dc.b    0,$0B,0,$0B,0,$8B,0,$8B     514
732                                             514
733 *                                           514
734 * deadable keys                             514
735 *                                           514
736                                             514
737 macro_a macro                               658
738 keydata_a:                                  339
739         dc.b
                DPF_MOD,keydata_a_alone-keyda
        ta_a                                    128
740         dc.b
                DPF_MOD,keydata_a_shift-keyda
        ta_a                                    228
741         dc.b
                0,'',0,'',0,$01,0,$01,0,$81
        ,0,$81                                  171
742 keydata_a_alone:                            974
743         dc.b    'a','á','à','','ã','ä'      876
744         ifne    KEYMAP_DOUBLE               135
745         dc.b
                'á','á','','á','á','á','à','
        ','à','à','à','à'                       679
746         endc                                629
747 keydata_a_shift:                            067
748         dc.b    'A','Á','À','Â','Ã','Ä'     721
749         ifne    KEYMAP_DOUBLE               372
750         dc.b
                'Á','Á','Â','Á','Á','Á','Á','
        Â','À','À','À','À'                      524
751         endc                                450
752         endm                                482
753                                             482
754 macro_e macro                               446
755 keydata_e:                                  763
756         dc.b
                DPF_MOD,keydata_e_alone-keyda
        ta_e                                    032
757         dc.b
                DPF_MOD,keydata_e_shift-keyda
        ta_e                                    116
758         dc.b
                0,'c',0,'c',0,$05,0,$05,0,$85
        ,0,$85                                  857
759 keydata_e_alone:                            888
760         dc.b    'e','é','è',','e','        ' 311
761         ifne    KEYMAP_DOUBLE               074
762         dc.b
                'é','é',','é','é','é','è','
       ','è','è','è','è'                       314
763         endc                                184
764 keydata_e_shift:                            218
765         dc.b    'E','É','È','Ê','E','Ë'     161
766         ifne    KEYMAP_DOUBLE               964
767         dc.b
                'É','É','Ê','É','É','É','È','
        Ê','È','È','È','È'                      852
768         endc                                906
769         endm                                018
770                                             018
771 macro_i macro                               530
772 keydata_i:                                  291
773         dc.b
                DPF_MOD,keydata_i_alone-keyda
        ta_i                                    016
774         dc.b
                DPF_MOD,keydata_i_shift-keyda
        ta_i                                    516
775         dc.b
                0,'¡',0,'¦',0,$09,0,$09,0,$89
        ,0,$89                                  818
776 keydata_i_alone:                            805
'777        dc.b    'i','!','

','i','
'   036
778         ifne    KEYMAP_DOUBLE               023
779         dc.b
                '!','!','

','!','!','!','

','          495
780         endc                                573
781 keydata_i_shift:                            331
782         dc.b    'I','Ì','Ì','Î','I','Ï'     042
783         ifne    KEYMAP_DOUBLE               349
784         dc.b
                'Ì','Ì','Î','Ì','Ì','Ì','Ì','
        Î','Ì','Ì','Ì','Ì'                      405
785         endc                                323
786         endm                                243
787                                             243
788 macro_n macro                               804
789 keydata_n:                                  646
790         dc.b
                DPF_MOD,keydata_n_alone-keyda
        ta_n                                    333
791         dc.b
                DPF_MOD,keydata_n_shift-keyda
        ta_n                                    203
792         dc.b
                0,' ',0,'D',0,$0E,0,$0E,0,$8E
        ,0,$8E                                  498
793 keydata_n_alone:                            758
794         dc.b    'n','n','n','n','ñ','n'     872
795         ifne    KEYMAP_DOUBLE               203
796         dc.b
                'n','n','n','n','n','n','n','
        n','n','n','n','n'                      422
797         endc                                484
798 keydata_n_shift:                            939
799         dc.b    'N','N','N','N','Ñ','N'     781
800         ifne    KEYMAP_DOUBLE               016
801         dc.b
                'N','N','N','N','N','N','N','
        N','N','N','N','N'                      787
802         endc                                929
803         endm                                561
804                                             561
805 macro_o macro                               547
806 keydata_o:                                  970
807         dc.b
                DPF_MOD,keydata_o_alone-keyda
        ta_o                                    435
808         dc.b
                DPF_MOD,keydata_o_shift-keyda
        ta_o                                    475
809         dc.b
                0,'m',0,'Ø',0,$0F,0,$0F,0,$8F
        ,0,$8F                                  386
810 keydata_o_alone:                            467
811         dc.b    'o','"','ò','','õ','ö'     649
812         ifne    KEYMAP_DOUBLE               980
813         dc.b
                '"','"','','"','"','"','ò','
        ','ò','ò','ò','ò'                       036
814         endc                                322
815 keydata_o_shift:                            494
816         dc.b    'O','Ó','Ò','Ô','Õ','Ö'     796
817         ifne    KEYMAP_DOUBLE               927
818         dc.b
                'Ó','Ó','Ô','Ó','Ó','Ó','Ò','
        Ô','Ò','Ò','Ò','Ò'                      015
819         endc                                605
820         endm                                661
821                                             661
822 macro_u macro                               333
823 keydata_u:                                  554
824         dc.b
                DPF_MOD,keydata_u_alone-keyda
        ta_u                                    647
825         dc.b
                DPF_MOD,keydata_u_shift-keyda
        ta_u                                    643
826         dc.b
                0,'f',0,'f',0,$15,0,$15,0,$95
        ,0,$95                                  100
827 keydata_u_alone:                            427
828         dc.b    'u','#','ù','','u','ü'     286
829         ifne    KEYMAP_DOUBLE               905
830         dc.b
                '#','#','','#','#','#','ù','
        ','ù','ù','ù','ù'                       293
831         endc                                467
832 keydata_u_shift:                            773
833         dc.b    'U','Ú','Ù','Û','U','Ü'     280
834         ifne    KEYMAP_DOUBLE               531
835         dc.b
                'Ú','Ú','Û','Ú','Ú','Ú','Ù','
        Û','Ù','Ù','Ù','Ù'                      847
836         endc                                413
837         endm                                493
838                                             493
839 macro_y macro                               529
840 keydata_y:                                  066
841         dc.b
                DPF_MOD,keydata_y_alone-keyda
        ta_y                                    767
842         dc.b
                DPF_MOD,keydata_y_shift-keyda
        ta_y                                    603
843         ifne    KEYMAP_USA1                 705
844         dc.b    0,'¤',0,'¥'                 541
845         endc                                611
846         ifne    KEYMAP_D                    111
847         dc.b    0,'q',0,'*'                 967
848         endc                                429
849         dc.b    0,$19,0,$19,0,$99,0,$99     320
850 keydata_y_alone:                            771
   658
852         ifne    KEYMAP_DOUBLE               517
853         dc.b
                'ý','ý','ý','ý','ý','ý','y','
        y','y','y','y','y'                      676
854         endc                                962
855 keydata_y_shift:                            584
856         dc.b    'Y','Ý','Y','Y','Y','Y'     249
857         ifne    KEYMAP_DOUBLE               988
858         dc.b
                'Ý','Ý','Ý','Ý','Ý','Ý','Y','
        Y','Y','Y','Y','Y'                      043
859         endc                                593
860         endm                                993
861                                             993
862 macro_space     macro                       556
863 keydata_space:                              320
864         dc.b
                DPF_MOD,keydata_space_alone-k
        eydata_space                            995
865         dc.b    0,$A0                       823
866 keydata_space_alone:                        013
867         dc.b    ' ','´','`','^','~','¨'     716
868         ifne    KEYMAP_DOUBLE               719
869         dc.b
                '´','´','^','´','´','´','`','
        ^','`','`','`','`'                      826
870         endc                                696
871         endm                                800
872                                             800
873         ifne    KEYMAP_USA1                 198
874         macro_a                             008
875         macro_e                             790
876         macro_i                             416
877         macro_n                             047
878         macro_o                             899
879         macro_u                             965
```




```
880         macro_y                            979
881         macro_space                        040
882         endc                               926
883                                            926
884         ifne    KEYMAP_D                   726
885         macro_e                            092
886         macro_u                            074
887         macro_i                            804
888         macro_o                            300
889         macro_a                            582
890         macro_n                            941
891         macro_y                            603
892         macro_space                        592
893         endc                               046
894                                            046
895 *                                          046
896 * miscellaneous string keys                046
897 *                                          046
898 * The first byte is the length of the
   string, the second                          046
899 * byte is the offset where the actual
   string data starts.                         046
900 *                                          046
901                                            046
902 keydata_tab:                               721
903         dc.b
              keydata_tab_alone_e-keydata_t
        ab_alone                               359
904         dc.b
              keydata_tab_alone-keydata_tab 115
905         dc.b
              keydata_tab_shift_e-keydata_t
        ab_shift                               231
906         dc.b
              keydata_tab_shift-keydata_tab 746
907 keydata_tab_alone:                         479
908         dc.b    $09                        515
909 keydata_tab_alone_e:                       128
910 keydata_tab_shift:                         960
911         dc.b    $9B,'Z'                    974
912 keydata_tab_shift_e:                       094
913                                            094
914 keydata_up:                                871
915         dc.b
              keydata_up_alone_e-keydata_up
        _alone                                 601
916         dc.b
              keydata_up_alone-keydata_up   297
917         dc.b
              keydata_up_shift_e-keydata_up
        _shift                                 461
918         dc.b
              keydata_up_shift-keydata_up   568
919 keydata_up_alone:                          443
920         dc.b    $9B,'A'                    020
921 keydata_up_alone_e:                        607
922 keydata_up_shift:                          697
923         dc.b    $9B,'T'                    657
924 keydata_up_shift_e:                        411
925                                            411
926 keydata_down:                              575
927         dc.b
              keydata_down_alone_e-keydata_
        down_alone                             459
928         dc.b
              keydata_down_alone-keydata_do
        wn                                     621
929         dc.b
              keydata_down_shift_e-keydata_
        down_shift                             963
930         dc.b
              keydata_down_shift-keydata_do
        wn                                     312
931 keydata_down_alone:                        162
932         dc.b    $9B,'B'                    192
933 keydata_down_alone_e:                      842
934 keydata_down_shift:                        967
935         dc.b    $9B,'S'                    818
936 keydata_down_shift_e:                      391
937                                            391
938 keydata_right:                             181
939         dc.b
              keydata_right_alone_e-keydata
        _right_alone                           201
940         dc.b
              keydata_right_alone-keydata_r
        ight                                   451
941         dc.b
              keydata_right_shift_e-keydata
        _right_shift                           077
942         dc.b
              keydata_right_shift-keydata_r
        ight                                   886
943 keydata_right_alone:                       614
944         dc.b    $9B,'C'                    361
945 keydata_right_alone_e:                     161
946 keydata_right_shift:                       964
947         dc.b    $9B,' @'                   876
948 keydata_right_shift_e:                     479
949                                            479
950 keydata_left:                              294
951         dc.b
              keydata_left_alone_e-keydata_
        left_alone                             716
952         dc.b
              keydata_left_alone-keydata_le
        ft                                     656
953         dc.b
              keydata_left_shift_e-keydata_
        left_shift                             368
954         dc.b
              keydata_left_shift-keydata_le
        ft                                     199
955 keydata_left_alone:                        336
956         dc.b    $9B,'D'                    832
957 keydata_left_alone_e:                      245
958 keydata_left_shift:                        473
959         dc.b    $9B,' A'                   802
960 keydata_left_shift_e:                      202
961                                            202
962 keydata_f1:                                169
963         dc.b
              keydata_f1_alone_e-keydata_f1
        _alone                                 911
964         dc.b
              keydata_f1_alone-keydata_f1   195
965         dc.b
              keydata_f1_shift_e-keydata_f1
        _shift                                 115
966         dc.b
              keydata_f1_shift-keydata_f1   858
967 keydata_f1_alone:                          811
968         dc.b    $9B,'0~'                   213
969 keydata_f1_alone_e:                        922
970 keydata_f1_shift:                          686
971         dc.b    $9B,'10~'                  389
972 keydata_f1_shift_e:                        557
973                                            557
974 keydata_f2:                                821
975         dc.b
              keydata_f2_alone_e-keydata_f2
        _alone                                 297
976         dc.b
              keydata_f2_alone-keydata_f2   683
977         dc.b
              keydata_f2_shift_e-keydata_f2
        _shift                                 769
978         dc.b
              keydata_f2_shift-keydata_f2   030
979 keydata_f2_alone:                          412
980         dc.b    $9B,'1~'                   067
981 keydata_f2_alone_e:                        461
982 keydata_f2_shift:                          130
983         dc.b    $9B,'11~'                  762
984 keydata_f2_shift_e:                        843
985                                            843
986 keydata_f3:                                120
987         dc.b
              keydata_f3_alone_e-keydata_f3
        _alone                                 034
988         dc.b
              keydata_f3_alone-keydata_f3   394
989         dc.b
              keydata_f3_shift_e-keydata_f3
        _shift                                 206
990         dc.b
              keydata_f3_shift-keydata_f3   953
991 keydata_f3_alone:                          928
992         dc.b    $9B,'2~'                   736
993 keydata_f3_alone_e:                        259
994 keydata_f3_shift:                          269
995         dc.b    $9B,'12~'                  118
996 keydata_f3_shift_e:                        292
997                                            292
998 keydata_f4:                                370
999         dc.b
              keydata_f4_alone_e-keydata_f4
        _alone                                 234
1000        dc.b
              keydata_f4_alone-keydata_f4   392
1001        dc.b
              keydata_f4_shift_e-keydata_f4
        _shift                                 842
1002        dc.b
              keydata_f4_shift-keydata_f4   899
1003 keydata_f4_alone:                         199
1004        dc.b    $9B,'3~'                   508
1005 keydata_f4_alone_e:                       252
1006 keydata_f4_shift:                         367
1007        dc.b    $9B,'13~'                  537
1008 keydata_f4_shift_e:                       624
1009                                           624
1010 keydata_f5:                               571
1011        dc.b
              keydata_f5_alone_e-keydata_f5
        _alone                                 737
1012        dc.b
              keydata_f5_alone-keydata_f5   613
1013        dc.b
              keydata_f5_shift_e-keydata_f5
        _shift                                 469
1014        dc.b
              keydata_f5_shift-keydata_f5   756
1015 keydata_f5_alone:                         017
1016        dc.b    $9B,'4~'                   199
1017 keydata_f5_alone_e:                       168
1018 keydata_f5_shift:                         912
1019        dc.b    $9B,'14~'                  467
1020 keydata_f5_shift_e:                       991
1021                                           991
1022 keydata_f6:                               835
1023        dc.b
              keydata_f6_alone_e-keydata_f6
        _alone                                 143
1024        dc.b
              keydata_f6_alone-keydata_f6   585
1025        dc.b
              keydata_f6_shift_e-keydata_f6
        _shift                                 015
1026        dc.b
              keydata_f6_shift-keydata_f6   356
1027 keydata_f6_alone:                         830
1028        dc.b    $9B,'5~'                   513
1029 keydata_f6_alone_e:                       407
1030 keydata_f6_shift:                         984
1031        dc.b    $9B,'15~'                  876
1032 keydata_f6_shift_e:                       729
1033                                           729
1034 keydata_f7:                               786
1035        dc.b
              keydata_f7_alone_e-keydata_f7
```

```
          _alone                                 356
1036      dc.b
             keydata_f7_alone-keydata_f7         260
1037      dc.b
             keydata_f7_shift_e-keydata_f7
          _shift                                 800
1038      dc.b
             keydata_f7_shift-keydata_f7         091
1039 keydata_f7_alone:                           006
1040      dc.b    $9B,'6~'                       058
1041 keydata_f7_alone_e:                         433
1042 keydata_f7_shift:                           271
1043      dc.b    $9B,'16~'                      692
1044 keydata_f7_shift_e:                         878
1045                                             878
1046 keydata_f8:                                 248
1047      dc.b
             keydata_f8_alone_e-keydata_f8
          _alone                                 712
1048      dc.b
             keydata_f8_alone-keydata_f8         950
1049      dc.b
             keydata_f8_shift_e-keydata_f8
          _shift                                 208
1050      dc.b
             keydata_f8_shift-keydata_f8         321
1051 keydata_f8_alone:                           641
1052      dc.b    $9B,'7~'                       882
1053 keydata_f8_alone_e:                         246
1054 keydata_f8_shift:                           821
1055      dc.b    $9B,'17~'                      915
1056 keydata_f8_shift_e:                         398
1057                                             398
1058 keydata_f9:                                 445
1059      dc.b
             keydata_f9_alone_e-keydata_f9
          _alone                                 571
1060      dc.b
             keydata_f9_alone-keydata_f9         015
1061      dc.b
             keydata_f9_shift_e-keydata_f9
          _shift                                 087
1062      dc.b
             keydata_f9_shift-keydata_f9         918
1063 keydata_f9_alone:                           119
1064      dc.b    $9B,'8~'                       593
1065 keydata_f9_alone_e:                         302
1066 keydata_f9_shift:                           882
1067      dc.b    $9B,'18~'                      241
1068 keydata_f9_shift_e:                         113
1069                                             113
1070 keydata_f10:                                452
1071      dc.b
             keydata_f10_alone_e-keydata_f
          10_alone                               754
1072      dc.b
             keydata_f10_alone-keydata_f10       014
1073      dc.b
             keydata_f10_shift_e-keydata_f
          10_shift                               602
1074      dc.b
             keydata_f10_shift-keydata_f10       661
1075 keydata_f10_alone:                          826
1076      dc.b    $9B,'9~'                       153
1077 keydata_f10_alone_e:                        126
1078 keydata_f10_shift:                          630
1079      dc.b    $9B,'19~'                      046
1080 keydata_f10_shift_e:                        118
1081                                             118
1082 keydata_help:                               247
1083      dc.b
             keydata_help_alone_e-keydata_
          help_alone                             001
1084      dc.b
             keydata_help_alone-keydata_he
          lp                                     161
1085 keydata_help_alone:                         936
1086      dc.b    $9B,'?~'                       861
1087 keydata_help_alone_e:                       252
1088                                             252
1089 keyMapName:                                 574
1090      ifne    KEYMAP_HEX                     779
1091      ifne    KEYMAP_USA1                    905
1092      dc.b    'usa1-hex',0                   130
1093      endc                                   128
1094      ifne    KEYMAP_D                       048
1095      dc.b    'd-hex',0                      527
1096      endc                                   117
1097      endc                                   571
1098      ifeq    KEYMAP_HEX                     599
1099      ifne    KEYMAP_USA1                    445
1100      dc.b    'usa1',0                       828
1101      endc                                   474
1102      ifne    KEYMAP_D                       034
1103      dc.b    'd',0                          279
1104      endc                                   021
1105      endc                                   987
1106      cnop    0,2                            329
1107                                             329
1108      end                                    036
```

Alle Geheimnisse der Tastatur offenbart dieses Listing dem Assemblerkenner. Vor dem Abtippen wird gewarnt!!

Programmname: d.hex
Sprache: Assembler, HEX

Auf Leserdiskette

HEX **Eingabe mit Nanomon**

```
000 00 00 03 f3 00 00 00 00 00 00 00 01 00 00 00 00 8b
010 00 00 00 00 00 00 01 4e 00 00 03 e9 00 00 01 4e 83
020 60 2c 00 00 00 00 00 00 00 00 00 00 05 30 00 00 7f
030 00 62 00 00 00 da 00 00 00 44 00 00 00 53 00 00 42
040 00 a2 00 00 01 da 00 00 00 4c 00 00 00 5b 2c 78 d8
050 00 04 93 c9 4e ae fe da 20 40 70 79 21 40 00 94 1b
060 70 ff 4e 75 00 00 ff 07 ff 07 fe 00 00 04 00 7c ca
070 00 00 00 ff bf ff ef ff ef ff f7 47 f4 ff 7f 00 50
080 00 00 07 03 03 03 03 03 07 03 03 03 03 07 23 07 2e
090 80 00 07 07 27 07 07 07 27 27 27 07 07 07 80 00 a1
0a0 00 00 27 07 07 27 27 27 27 27 07 03 03 07 80 00 26
0b0 00 00 01 27 07 07 07 07 27 07 03 03 07 80 00 00 56
0c0 00 00 22 00 41 00 04 02 00 80 80 80 01 80 41 41 84
0d0 41 41 41 41 41 41 41 41 41 41 41 41 07 07 01 01 dd
0e0 01 40 80 80 80 80 80 80 80 80 80 80 80 80 80 80 5b
0f0 80 80 80 80 80 80 80 80 80 80 7e 60 7e 60 21 b9 b5
100 21 31 b2 40 22 32 23 b3 a7 33 a2 b0 24 34 25 bc 44
110 25 35 5e bd 26 36 26 be 2f 37 2a b7 28 38 28 ab e5
120 29 39 29 bb 3d 30 5f 2d 3f df 00 00 02 ba 7c 5c 50
130 7c 5c 00 00 00 00 00 00 00 30 c5 e5 51 71 b0 b0 8c
140 57 77 00 00 03 12 ae ae 52 72 de fe 54 74 a5 a4 41
150 5a 7a 00 00 03 46 00 00 03 7a 00 00 03 ae b6 b6 35
160 50 70 7b 5b dc fc 7d 5d 2a 2b 00 00 00 00 00 00 64
170 00 31 00 00 00 32 00 00 00 33 00 00 03 e2 a7 df 35
180 53 73 d0 f0 44 64 00 00 02 c2 00 00 02 d2 00 00 9b
190 02 e2 00 00 02 f2 00 00 03 02 a3 a3 4c 6c 3a 3b e6
1a0 d6 f6 22 27 c4 e4 5e 23 5e 23 00 00 00 00 00 00 a2
1b0 00 34 00 00 00 35 00 00 00 36 00 00 3e 3c 00 00 21
1c0 04 4a f7 d7 58 78 c7 e7 43 63 aa aa 56 76 ba ba 7d
1d0 42 62 00 00 04 16 bf b8 4d 6d 3c 2c 3b 2c 3e 2e 48
1e0 3a 2e 3f 2f 5f 2d 00 00 00 00 00 00 00 2e 00 00 0f
1f0 00 37 00 00 00 38 00 00 00 39 00 00 04 7e 00 00 76
200 00 08 00 00 04 94 00 00 00 0d 00 00 0a 0d 00 00 0e
210 9b 1b 00 00 00 7f 00 00 00 00 00 00 00 00 00 00 70
220 00 00 00 00 65 2d 00 00 00 00 00 00 04 9b 00 00 11
230 04 a3 00 00 04 ab 00 00 04 b4 00 00 04 bd 00 00 a0
240 04 c8 00 00 04 d3 00 00 04 de 00 00 04 e9 00 00 28
250 04 f4 00 00 04 ff 00 00 05 0a 00 00 05 15 00 00 2f
260 05 20 7b 7b 61 5b 7d 7d 62 5d 00 00 63 2f 00 00 16
270 64 2a 00 00 66 2b 00 00 05 2b 00 00 00 00 00 00 78
280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 5a
290 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b7
2a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 14
2b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 71
2c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ce
2d0 00 00 00 00 00 00 00 00 00 00 08 61 08 62 00 3d 9b
2e0 00 2b 00 66 00 46 08 61 08 61 00 06 00 06 00 86 83
2f0 00 86 00 67 00 47 08 62 08 62 00 07 00 07 00 87 92
300 00 87 00 68 00 48 08 03 08 03 00 08 00 08 00 88 80
310 00 88 00 6a 00 4a 08 04 08 04 00 0a 00 0a 00 8a fa
320 00 8a 00 6b 00 4b 08 05 08 05 00 0b 00 0b 00 8b 98
330 00 8b 01 10 01 22 00 a9 00 a9 00 05 00 05 00 85 9a
340 00 85 65 e9 e8 ea 65 eb e9 e9 ea e9 e9 e9 e8 ea fd
350 e8 e8 e8 e8 45 c9 c8 ca 45 cb c9 c9 ca c9 c9 c9 33
360 c8 ca c8 c8 c8 c8 01 10 01 22 00 b5 00 b5 00 15 a6
370 00 15 00 95 00 95 75 fa f9 fb 75 fc fa fa fb fa f2
380 fa fa f9 fb f9 f9 f9 f9 55 da d9 db 55 dc da da 7f
390 db da da da d9 db d9 d9 d9 d9 01 10 01 22 00 a1 ae
3a0 00 a6 00 09 00 09 00 89 00 89 69 ed ec ee 69 ef 97
3b0 ed ed ee ed ed ed ec ee ec ec ec ec 49 cd cc ce 1f
3c0 49 cf cd cd ce cd cd cd cc ce cc cc cc cc 01 10 d9
3d0 01 22 00 f8 00 d8 00 0f 00 0f 00 8f 00 8f 6f f3 8d
3e0 f2 f4 f5 f6 f3 f3 f4 f3 f3 f3 f2 f4 f2 f2 f2 f2 c7
3f0 4f d3 d2 d4 d5 d6 d3 d3 d4 d3 d3 d3 d2 d4 d2 d2 fe
400 d2 d2 01 10 01 22 00 e6 00 c6 00 01 00 01 00 81 6b
410 00 81 61 e1 e0 e2 e3 e4 e1 e1 e2 e1 e1 e1 e0 e2 4e
420 e0 e0 e0 e0 41 c1 c0 c2 c3 c4 c1 c1 c2 c1 c1 c1 c0
430 c0 c2 c0 c0 c0 c0 01 10 01 22 00 ad 00 af 00 0e 73
440 00 0e 00 8e 00 8e 6e 6e 6e 6e f1 6e 6e 6e 6e 6e cd
450 6e 6e 6e 6e 6e 6e 6e 6e 4e 4e 4e 4e d1 4e 4e 4e 94
460 4e 4e 4e 4e 4e 4e 4e 4e 4e 4e 01 10 01 22 00 b1 2b
470 00 ac 00 19 00 19 00 99 00 99 79 fd 79 79 79 ff 23
480 fd fd fd fd fd fd 79 79 79 79 79 79 59 dd 59 59 c8
490 59 59 dd dd dd dd dd dd 59 59 59 59 59 59 01 04 74
4a0 00 a0 20 b4 60 5e 7e a8 b4 b4 5e b4 b4 b4 60 5e be
4b0 60 60 60 60 01 04 02 05 09 9b 5a 02 04 02 06 9b ae
4c0 41 9b 54 02 04 02 06 9b 42 9b 53 02 04 03 06 9b 4b
4d0 43 9b 20 40 02 04 03 06 9b 44 9b 20 41 03 04 04 4c
4e0 07 9b 30 7e 9b 31 30 7e 03 04 04 07 9b 31 7e 9b 5b
4f0 31 31 7e 03 04 04 07 9b 32 7e 9b 31 32 7e 03 04 67
500 04 07 9b 33 7e 9b 31 33 7e 03 04 04 07 9b 34 7e ac
510 9b 31 34 7e 03 04 04 07 9b 35 7e 9b 31 35 7e 03 5a
520 04 04 07 9b 36 7e 9b 31 36 7e 03 04 04 07 9b 37 d1
530 7e 9b 31 37 7e 03 04 04 07 9b 38 7e 9b 31 38 7e 0c
540 03 04 04 07 9b 39 7e 9b 31 39 7e 03 02 9b 3f 7e 75
550 64 2d 68 65 78 00 00 00 00 00 03 ec 00 00 00 27 e6
560 00 00 00 00 00 00 02 56 00 00 02 3e 00 00 02 3a 67
570 00 00 02 36 00 00 02 32 00 00 02 2e 00 00 02 2a 50
580 00 00 02 26 00 00 02 22 00 00 02 1e 00 00 02 1a 6d
590 00 00 02 16 00 00 02 12 00 00 02 0e 00 00 02 0a 8a
5a0 00 00 01 e2 00 00 01 da 00 00 01 b2 00 00 01 9e 1f
5b0 00 00 01 76 00 00 01 72 00 00 01 6e 00 00 01 6a b0
5c0 00 00 01 66 00 00 01 5a 00 00 01 3a 00 00 01 36 3d
5d0 00 00 01 32 00 00 01 22 00 00 01 0a 00 00 00 2a 25
5e0 00 00 00 26 00 00 00 22 00 00 00 1e 00 00 00 1a 73
5f0 00 00 00 16 00 00 00 12 00 00 00 0e 00 00 00 0a 90
600 00 00 00 00 00 00 03 f2                         af
```



Nach Abtippen mit Nanomon wird das Programm als „d.hex“ bei den anderen Keymaps gespeichert. Der Befehl „SetMap d.hex“ belegt nun den Zehnerblock im HEX-Format.

★ **S·A·F·E** ★

SALZBURGER AMIGA FREUNDESKREIS
SOFTWARE MADE IN AUSTRIA

01	DRUCKER TREIBER	16	VIDEO-TITEL VERW.
02	DISC-CAT	17	ÖSTERR. LOTTO
03	AMIGAMES	18	ZIMMER VERWALTUNG
04	ICON-JOY	19	FONT-DISC
05	C-BEFEHLE	19a	FONTS
06	SHANGHAI	20	SOUND-DISC
07	UTILITIES	20a	SOUND-FILES
08	PICTURES 1	21	ANTI ATARI
09	PICTURES 2	22	NEW GAMES
10	BACKGAMMON	23	SOFTLANDING
11	MILESTONE	24	DISCOVERY
12	MONOPOLI	25	TERMINAL-DISC
13	JETZT SCHLÄGT'S 13	26	LOTTOMANIA
14	CRIBBAGE	27	ANTI VIRUS DISC
15	OTHELLO	28	AMIGABUCH

! ACHTUNG PROGRAMMIERER !
WIR SUCHEN STÄNDIG LEUTE, DIE SPEZIELL FÜR DEN EUROPÄISCHEN MARKT PROGRAMME ERSTELLEN KÖNNEN.

FISH · FAUG · RPD · PANORAMA · AMICUS
AMISOFT · CASA · T BAG · AUSTRIA · AUGE
UND VIELES MEHR. ÜBER 500 DISKETTEN
KATALOG (3 DISK.) ÖS 90,- PD DISKETTE AB ÖS 40,-

IHRE PD-SPEZIALISTEN IN ÖSTERREICH

BERND KÜPPERS
M. AICHBERGER

A-5730 MITTERSILL, FELBERSTRASSE 7 TEL.: 06562/282 OD. 4073

AMIGA 500/1000/2000

	DM		DM
IC 8362	39,10	NETZTEIL A-500	153,33
IC 8364	58,48	MAUS A-500/2000	106,70
IC 8520	29,98	TASTATUR A-500	209,08
KICKST. V1.2	23,60	VIDEO-HYBRID A-500	23,60
IC 68000-8	19,49	TASTATUR-KABEL A-1000	30,55
IC 8371 FAT	80,03	TASTATUR A-1000	311,22
IC 5719 GARY	23,60	NETZTEIL A-2000	414,85
IC 6242 B	17,90	TASTATUR A-2000	297,54
IC 6570-036	29,75	QUARTZ 28.37516 NTSC	23,60
IC 8367 PAL	94,28	QUARTZ 28.63636 PAL	12,31
IC 8361 NTSC	60,76	DRAM 41256-15	19,95

externes Tastaturgehäuse für A-500 117,08
passendes Verbindungskabel hierzu 38,42

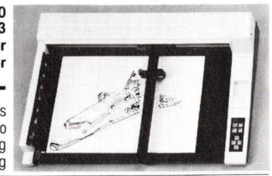
sowie (fast) alle CBM-Chips für Reparatur + Service
Komplettes Lieferprogramm gegen DM 2,50 in Briefmarken.
HÄNDLERANFRAGEN ERWÜNSCHT
(bitte Nachweis beifügen)

CIK-Computertechnik — Ingo Klepsch
Postfach 13 31 5828 Ennepetal 1
Tel. 0 23 33/8 02 02 Fax 0 23 33/7 03 45

Technocom®

DIN-A3-Flachbettplotter

Sekonic SPL-450 sehr schneller A3 HP-GL-kompatibler Plotter



1998,-
Lieferung: frei Haus
inkl.: Porto
Verpackung
Transportversicherung

Technische Daten:		**Zubehör:**
Type:	Flachbett	Kabel
Papiermaße:	A3 u. kleiner	8 Farbstifte
Plottbereich:	385 × 280 mm	Handbuch
Geschwindigkeit:	400 mm/Sek.	Staubschutzhaube
Auflösung:	0,025 mm	
Wiederholgenauigkeit:	0,2 mm	
Stiftwechselgenauigkeit:	0,3 mm	
Anzahl der Stifte:	8 mit autom. Verschlußsystem	
Schnittstellen:	Centronics, parallel	
	RS232C, seriell	HP-GL ist ein-
Plottbefehle:	56 HP*-GL-kompatibel	getragenes Waren-
Stifte:	alle Staedler- oder	zeichen der
	HP*-Plotterstifte	Hewlett-Packard

Telefon: (089) 4 30 37 02

Technocom
Jochy Polierer
Wasserburger Landstraße 180
D-8000 München 82

telefonische Bestellungen:
Mo.—Fr.: 14—22 Uhr
Sa.: 9—14 Uhr

AMIGA - SOFTWARE

Public Domain Disketten

Jetzt über 700 Public Domain Disketten vorrätig von ACS, Amicus, AUGE, Chiron Conception, Faug, Fish, Kickstart, Panorama, RPD, SACC, Software Digest, TBAG und Tornado.

Einzeldiskette 3,50 DM

Paketangebote:
10 Disketten Ihrer Wahl **33,00 DM**
20 Disketten Ihrer Wahl **65,00 DM**
Gesamtkataolg 6,00 DM
Für unsere Public Domain Software verwenden wir 2DD **Qualitätsdisketten.**
Obige Preise zuzüglich Versandkosten.
Speichererweiterung 512 KB für Amiga 500 lieferbar.
Aufpreis für **2 DD Markendisketten 1,30 DM** pro Diskette.
Zubehör:
Diskettenlaufwerk mit NEC 1037a anschlußfertig für alle Amiga-Modelle **328 DM,** Speichererweiterung 512 KB für Amiga 500, abschaltbar, **mit Uhr 239 DM**

A. Fischer, Kirchstr. 40, Tel. 05257- 4347
4794 Hövelhof

HK-COMPUTER

● Hardware ● Software ● EDV-Zubehör ● Beratung ● Service ●

Festplatten:
20 MB für A2000 intern 798,—
20 MB für A500/1000 komplett 948,—
Laufwerke:
3½" Rohlaufwerk 199,—
3½" A 2000 intern mit EinbauKit 219,—
3½" A 500/2000 extern anschlußfertig 269,—
abschaltbar, Bus durchgeschleift, AMIGAfarbenes Metallgehäuse
5¼" A 500/2000 extern anschlußfertig 319,—
abschaltbar, Bus durchgeschleift, AMIGAfarbenes Metallgeh., 40/80 Tracks, AMIGA & MS-DOS
Disketten 3½":
NoName 2S2D 10 Stück 22,90
NoName 2S2D 100 Stück 225,—
Verbatim VEREX 2S2D 10 Stück 28,—
Verbatim VEREX 2S2D 100 Stück 265,—
Diskettenbox für ca. 80 Disketten 16,50
Reinigungsset für 3½"-Laufwerke 9,—
Drucker und Zubehör:
Wir sind NEC-Vertriebspartner, alle NEC-Drucker mit deutschem Handbuch und 1 Jahr Garantie, keine Grauimporte! NEC-Drucker und Monitore zu interessanten Preisen, z. B.:
NEC P2200 898,—
Der neue NEC P6 Plus 1598,—
Druckerkabel in Profi-Qualität 2m = 15,— 5m = 29,—
RAM-Erweiterung 512 K mit Uhr, abschaltbar, Tagespreis
14"-Flat-Screen-Monitor 269,—
s/w, hochauflösend, Dreh-/Schwenkfuß, invers schaltbar, inkl. Amiga-Anschlußkabel
GO AMIGA! Text für Schüler/Studenten 199,—
BOOT-Selector alle AMIGA 14,50

HK-Computer Thomas Küpper, Mo—Fr 10—18 Uhr, Sa 10—14 Uhr
An der Wallburg 2, 5060 Berg. Gladbach 1, 0 22 04/2 21 24
und Overstolzenstraße 10, 5000 Köln 1, 02 21/31 16 06

UPS-Versand: Nachnahme 8 DM, Vorauskasse 4 DM, Großgeräte nach UPS-Tabelle ohne Aufschlag. Ausland nur Vorauskasse + 10 DM.
Fordern Sie unser kostenloses Info an.

Technocom®

Amiga 2000

B-Version
1-MB-Grundspeicher
— mit Farbmonitor 1084
— mit PC-Karte
— mit 5¼"-Laufwerk

2998,-
inkl.: Porto und Verpackung Lieferung: frei Haus!

Telefon: (089) 4 30 37 02

Technocom
Jochy Polierer
Wasserburger Landstraße 180
D-8000 München 82

telefonische Bestellungen:
Mo.—Fr.: 14—22 Uhr
Sa.: 9—14 Uhr

20 MB auf Floppy sichern

? ? ? ? ?

mit dem Sicherungsprogramm
cpio - Amiga
kein Problem!

Sichert 20 MB in 30 Minuten!
Sichert fortlaufend auf mehrere Disketten!
Sichert Dateien mit mehr als 880 KB!
Ein Programm aus Deutschland mit deutscher Anleitung und deutschem Dialog!

Preis: DM 79,- zuzügl. Versand

Beschreibung gegen Rückporto bei
Kurt Harders
Systemanalyse und Beratung
Angermunder Straße 252
4100 Duisburg 29
Tel. 0203/765377

HiTex-Editor V2.0

Der schnellste Amiga-Editor

- geeignet für alle Sprachen
- Workbench- & CLI-fähig
- maus- oder tastaturgesteuert
- leistungsfähige Batchsprache
- bearbeitet 10 Texte gleichzeitig
- 30 bzw. 62 Textzeilen sichtbar
- Blocklegen per Maus
- umfangreiche Blockoperationen
- extrem schnell: ∅ 2000 Ersetzungen/pro Sekunde
- Bildschirmaufbau in 0.037 Sekunden
- vollständig kompatibel
- ausführliche, deutsche Anleitung

Preis: DM 79,—

Info gegen Freiumschlag

Dietmar Heidrich
Blenderstraße 60
2800 BREMEN 61

Technocom®

Star LC 2410

24 Nadeln / 142 cps Pica
NLQ und Bit-Image Modus
7 KByte Druckspeicher
14 interne Zeichensätze
50 Download Charakter in 24N-Auflösung
10 LED indicator
Parallel Interface Centronics
220 V / 6,4 kg
Maße H 120 x B 410 x T 328 mm

998,00 DM

Telefon: (089) 4 30 37 02

Technocom
Jochy Polierer
Wasserburger Landstraße 180
D-8000 München 82

telefonische Bestellungen:
Mo.—Fr.: 14—22 Uhr
Sa.: 9—14 Uhr

Staubschutzhauben–

ein Problem weniger

z. B. für:
Amiga 500/2000 ....DM 19,—
Monitorhaube ....DM 29,—
Set-Preis .... **nur** DM 40,—
NEC-P6 Drucker
ohne Papiereinzug ....DM 19,—
mit Papiereinzug ....DM 25,—
Star-NL10 DM 19,—

Alle Preise zzgl. DM 5,50 (Versandkosten).
Fordern Sie unsere Gesamtpreisliste mit Materialmuster an!

K&S-Versand

K. Schellhammer, Dorfstraße 26
2061 Klein Wesenberg
Fax 04533/5287

Tel.: 0 45 33/35 66

C. V. S. – Versand

Zubehör für Amiga

Laufwerke extern
CHINON DRIVE 3,5" abschaltbar ....260,00 DM
CHINON DRIVE 5,25" 40/80 TR ....300,00 DM
Drucker/Scanner
NEC 2200 Pinwriter (24 Nadeln) ....898,00 DM
Präsident-Printer AMIGA-Zeichensatz,
Centronics-Schnittstelle ....399,00 DM
Handy-Scanner mit 16 Graustufen, inkl.
Grafikpaket, von Cameron ....858,00 DM
Neu: jetzt mit Texterkennungssoftware
Monitore
Schwarz/Weiß-Monitor, anschlußfertig
für den AMIGA, 20 MHz ....230,00 DM
Farbmonitor Sanyo ....648,00 DM
NEU: 20 MByte Harddisk für AMIGA ext.
komplett anschlußfertig ....989,00 DM
Software
AMIGA-TOOLS neue Utility-Disk mit Copy,
Viruskiller, RAM-Deleter etc ....49,95 DM
PD (sehr große Auswahl) ....ab 3,50 DM
über 1000 Disketten auch IBM-PD auf 3,5"
für Amiga-Transformer/PC-Karte/Sidecar
Katalogdisk (3 Disk) ....9,00 DM

C.V.S., Gutenbergstr. 5, 2300 Kiel 1
Tel. Bestellannahme: (04 31) 55 15 15
Aktuelle Preisliste auf Anforderung
Preisänderungen vorbehalten

KM Amiga-PD

- Einsteigerpaket 43,-DM
10 Disketten voll Info's und Hilfen für AMIGA-Anfänger.
- Spielepaket 43,-DM
Mehr als 40 Spiele auf 10 Disk.
- Das große PD-Buch I + II
Buch+10 Programmdisketten 90,-DM
- ca. 900 weitere Disketten
RPD, Fish, Faug, Panorama, etc....
Jede Diskette nur 4,50DM

Preise zzgl. Versandspesen.
2 Katalog-Disk gegen 5,-DM Vorkasse.
Schnellversand 1 Tag nach Auftrag.

Kirschbaum
Medienberatung
Schubertstr. 3 , 4320 Hattingen
Tel: 02324 / 82249

Krankenpfleger - Arzthelfer
Psychologen - Masseure - Heilpraktiker
Apothekenhelfer

Eine typische *AMIGA* Lösung:

MED-LERNPROGRAMM

Kompletter Medizin Kurs

12 Disketten Anatomie - Inhalt je Diskette:
* Anatomie
* Physiologie
* Grafik-Datei (Animationsgrafiken)
* Lern-Lexikon
* Testfragen-Datei

6 Disketten Pathologie - Inhalt je Diskette:
* Allgemeine Pathologie
* Spezielle Pathologie
* Lern-Lexikon
* Praxis Fallstudien

Gesamtpreis	DM	1650.–
Einzeldiskette	DM	99.–
Demo-Diskette	DM	10.–

proLinea

Potsdamer Str. 102
1000 Berlin 30
Tel.: 030 - 262 28 82

Musik- und Grafiksoftware Shop

Wasserburger Landstr. 244 ★ 8000 München 82
Telefon: 089/4306207

Umfangreiches Musik- und Midisoftware-Angebot für alle Amiga

Midi-Editorsoftware für folgende Synthesizer:
DX7/II, DX21, DX27, DX100, TF1, TX7, DX9, TX802, MT32, D50, ESQ1, ESQM, EX 80, Casio CZ 101, CZ 1000, CZ 3000, CZ 5000

Zahlreiche Sequenzerprogramme für AMIGA jetzt lieferbar!

MIDI-Interfaces für AMIGA 500/1000/2000 in verschiedenen Versionen.

Fordern Sie unsere kostenlose AMIGA-MIDI-LISTE an. Dort finden Sie Beschreibungen und Preise zu all unseren AMIGA-MIDI-Produkten.

Genlock, RGB Splitter, RGB-PAL-Wandler jetzt lieferbar

Alle gängigen Grafikprogramme, Soundsampler, Videodigitizer, Scanner und Grafiktableaus für Amiga auf Lager. Fordern Sie unsere kostenlose AMIGA-Preisliste an. Versand per Vorkasse oder Nachnahme.

Rufen Sie uns einfach an oder besuchen Sie uns in unserem Laden!
MO-FR 10-18.30 UHR ★ SA 9-13.00 UHR

PROFIBU

Finanzbuchhaltung
und

PROFAKT

Fakturierung

Ausgereifte, doppelte Finanzbuchhaltung und abgestimmte Fakturierung für

AMIGA oder **MS/DOS**

PROFIBU: Frei definierbarer Kontenrahmen, schnelle und einfache Handhabung, autom. Abschlußmöglichkeit, Steuersplitting, Storno, Skonto, umfangreiche Auswertungen und Ausdrucke.

PROFAKT: Artikel, Lager, Kunden, Lieferanten, Sofortfakturierung, Bestellung, Gutschrift, Angebot, Lieferschein, Rechnung, Mahnung, Offene Posten, umfangreiche Auswertungslisten, Daten an PROFIBU.

Bitte fordern Sie Informationsmaterial an!

Entwicklung und Vertrieb:
Sven Tietjen
Rigaweg 1, 3300 Braunschweig, 05309/1466

Achtung!

1. Software darf nur vom Berechtigten vervielfältigt und verbreitet werden. Originalprogramme sind am Originalaufkleber des Datenträgers (Diskette oder Kassette) und ggf. am Copyright-Vermerk zu erkennen. Auch befinden sie sich normalerweise in der Originalverpackung. Wir empfehlen Käufern dringend, darauf zu achten, daß sie tatsächlich Originalprogramme erwerben.
2. Es verstößt gegen das Urheberrecht bzw. das Gesetz gegen den unlauteren Wettbewerb, Raubkopien herzustellen oder zu verbreiten. Verstöße können zivilrechtlich und ggf. auch strafrechtlich geahndet werden. Käufern von Raubkopien steht kein Nutzungsrecht zu. Sie müssen mit Unterlassungsverfahren und ggf. mit sofortiger Beschlagnahme der erworbenen Raubkopien rechnen.
Wir erwarten von unseren Inserenten und Lesern, daß sie **keine** Raubkopien verbreiten, anbieten oder erwerben.
3. Anzeigen, die vermuten lassen, daß mit Raubkopien gehandelt wird, werden nicht veröffentlicht.

IHR COMPUTERAUSDRUCK ZUM AUFBÜGELN

MIT UNSEREM SPEZIALFARBBAND!!! NEU

STAR NL/NG 10 37,50
EPSON FX/RX 80 34,50
NEC P6 39,50

ACHTUNG!!

NEC CP6 4-FARB NORMAL........37,90

COMPEDO

PENNEKAMP-DORSCH Computerzubehör GbR

R. PENNEKAMP
POSTFACH 1352
5860 ISERLOHN
TEL.: 02371-29785

A. DORSCH
POSTFACH 100105
4630 BOCHUM
TEL.: 0234-12664

CPS

CPS Computertechnik GmbH
Marienstr. 16 · 3300 Braunschweig
Telefax (0531) 796461
BTX *20088 1490#
Tel. (0531) 794087

AMIGA
AMIGA 500 900,-
AMIGA 2000 1955,-
AMIGA 500 + PAL Modulator 955,-
AMIGA 500 + Phil. Mon. CM 8833 1485,-
AMIGA 500 + Commodore 1084 1408,-
Amiga 500 + CM 8833 +
Epson LX 800 o. STAR LC 10 2184,-
AMIGA 500 + CBM 1084 + EPSON LX 800
oder STAR LC 10 2087,-
AMIGA 2000 + 20 MB Harddisc,
SCSI Contr. 3079,-
AMIGA 2000 + Monitor 1084 2480,-
AMIGA 2000 + 1084 + PC/XT-Karte *
mit 5 ,25"-Laufwerk komplett 3148,-

DISKETTEN
NN 2DD 3,5" 10 Stck. 23,-
NN 2DD 5,25" 10 Stck. 9,-
FUJI MF2DD Double sided 135 TPI 10 Stck. ...29,-
NASHUA 2DD Double sided 96 TPI 10 Stck. ...19,-

DRUCKER
OKI Microline 390 1498,-
NEC P6 + ...1798,-
STAR LC 10 Color 875,-
EPSON LX-800, Centr. 699,-
STAR LC-10, Centr. 699,-
NEC P 2200, Centr. 898,-
Alle Drucker m. dt. Handbuch u.Seriennummer!

Angebote freibleibend, Zwischenverkauf vorbehalten. Wir liefern ausschließlich per UPS - Nachnahme, in der Regel innerhalb 24 Stunden.

CPS - weil Preis + Leistung stimmen!

AMIGA WELT INFOMARKT

Achtung!

1. Software darf nur vom Berechtigten vervielfältigt und verbreitet werden. Originalprogramme sind am Originalaufkleber des Datenträgers (Diskette oder Kassette) und ggf. am Copyright-Vermerk zu erkennen. Auch befinden sie sich normalerweise in der Originalverpackung. Wir empfehlen Käufern dringend, darauf zu achten, daß sie tatsächlich Originalprogramme erwerben.
2. Es verstößt gegen das Urheberrecht bzw. das Gesetz gegen den unlauteren Wettbewerb, Raubkopien herzustellen oder zu verbreiten. Verstöße können zivilrechtlich und ggf. auch strafrechtlich geahndet werden. Käufern von Raubkopien steht kein Nutzungsrecht zu. Sie müssen mit Unterlassungsverfahren und ggf. mit sofortiger Beschlagnahme der erworbenen Raubkopien rechnen.
   Wir erwarten von unseren Inserenten und Lesern, daß sie **keine** Raubkopien verbreiten, anbieten oder erwerben.
3. Anzeigen, die vermuten lassen, daß mit Raubkopien gehandelt wird, werden nicht veröffentlicht.

AMIGA WELT KLEINANZEIGEN

Zählt für Sie jeder Millimeter? Dann liegen Sie hier richtig mit Ihrem Angebot. Nutzen Sie diese Rubrik, um Ihre gewerbliche oder private Kleinanzeige zu plazieren. Wir berechnen DM 7,— zzgl ges. MwSt. pro Zeile für gewerbliche Anzeigen (1 Zeile = 33 Anschläge) und DM 5,— pro 5 Zeilen für private Anzeigen.

Und so wird es gemacht! Schicken Sie uns Ihren Text mit der genauen Adresse und nennen Sie uns die entsprechende Rubrik, unter der Ihre Anzeige erscheinen soll. Weitere Informationen geben wir Ihnen aber auch gerne unter Tel. 089/3 60 86-201.

GEWERBLICHE KLEINANZEIGEN

ACHTUNG AMIGA-USER ACHTUNG
Wir führen Software für Ihren Computer. Fordern Sie unsere kostenlose Liste an.
BERLAU-SOFT, Postfach 14 15
2150 Buxtehude.

Master 3A 3½'' Microfloppy von OCEANIC ELECTRONICS für Amiga 500, 1000, 2000: **DM 289,— (inkl. Porto).**
TAI TECH computerhandel,
W. Gramlich, Landeckstr. 12,
6740 Landau

★ Lohn-Einkommensteuer 1988 ★ vom Fachmann. Berechnet alles. Umfangreiche Erläuterungen, Tips. AMIGA: 79 DM; Info 1.10. Dipl.-Fin-Wirt U. Olufs, Bachstr. 70m, 5216 Niederkassel 2, 02208/4815

!! Preiswerte Spitzensoftware !! Amiga Public Domain. Alle üblichen Serien (Fish, Tbag...) sowie Grafik und Sound-Sonderserien. Pro Disk (2DD) ab 4,70. Katalog gegen DM 5,— in bar oder Briefm. Flight-II-Anleitung in Deutsch + Karten DM 20. Chris Schumacher, Wormser Str. 9, 5000 Köln 1, Tel.: 02 21/37 23 91

Gratisliste über Public-Domain für Amiga anfordern bei F. Neuper, 8473 Pfreimd, Postfach 72

Bisher nur für Super-Mini
und Unix-Systeme
JETZT AUCH FÜR AMIGA 2000
kompl. Finanzbuchhaltung
ab DM 4000,—
kompl. Auftragsabwicklung
ab DM 6000,—
Lagerwesen
ab DM 3000,—
Kosmo Data GmbH, Südstr. 16
5600 Wuppertal 1
Tel. 02 02/30 40 11

PRIVATE KLEINANZEIGEN

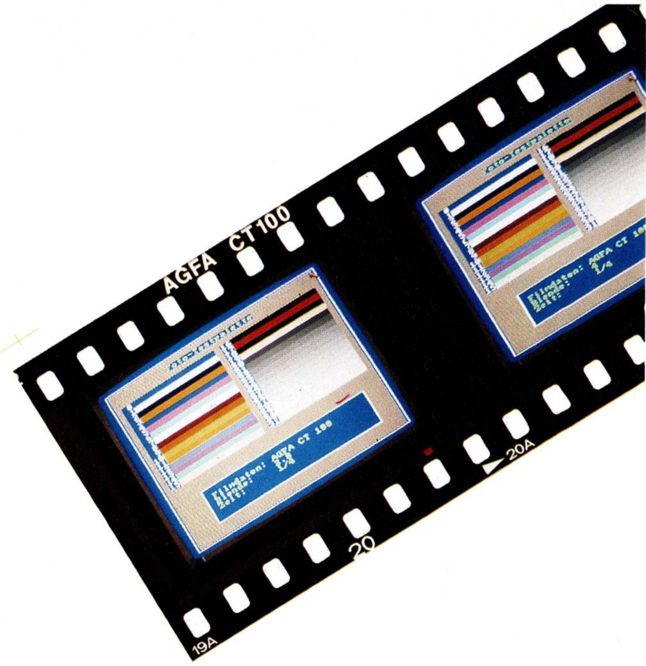
An alle professionellen Amiga-Anwender: Falls Sie Ihren Amiga professionell nutzen, sollten Sie unsere Info-Broschüre über die GPA anfordern. GPA, Hüttnerstr. 24, CH-8824 SCHÖNENBERG

Welcher Umsteiger gibt preiswert seine alte 64/128er Hard- u. Software, Bücher und Zeitschriften zum Einstieg für meinen Sohn ab? Tel. 0 24 02/8 18 64

■■■ Suche gebraucht: ■■■ PAL-Video-Genlock, Videodigitizer und RGB-Splitter. Zus. Org.-Soft.: Videoscape 3D PAL, Deluxe-Photolab. Angebot tel. ab 18.00 h: 0 71 33/1 57 94

Verkaufe Super Soft wie 20 000 Meilen unter dem Meer, Swooper, Adventure Construction Set usw., tausche auch zwei Titel Ihrer Wahl gegen Oblitevator oder Ooze. Call: 09 11/56 57 89

AMIGA-GRAFIK. Wir drucken für Sie in prof. Qualität m. **CALCOMP PAINTMASTER** o. **XEROX 4020** in brillanten Farben auf Papier o. Overhead-Folie f. Privat o. Präsentationen. Infos Tel. 0251/62214



BASIC-Testbild zum optimalen Bildschirmfoto

In der vorigen Ausgabe wurden Bildschirmfotografen in die Technik der Zelluloid-Hardcopy eingewiesen. An dieser Stelle liefern wir das versprochene Listing.

Programmname: Testbild
Sprache: AmigaBASIC

Auf Leserdiskette

ABC **Eingabe mit beliebigem Editor (ohne Zeilennummern) Überprüfen mit Checksum**

```
 1 REM ******* T E S T B I L D *******                 263
 2 SCREEN 1,320,256,5,1                               868
 3 WINDOW 2,,(0,0)-(311,230),16,1                     963
 4 ON BREAK GOSUB beenden:BREAK ON                    774
 5 COLOR 1,0:CLS:GOSUB Automatikdaten                 895
 6      ' Bildschirm mit Muster fuellen               896
 7 DIM pfeld%(3)                                      010
 8 pfeld%(0)= &HCCCC:pfeld%(1)= &H3333                753
 9 pfeld%(2)= &HCCCC:pfeld%(3)= &H3333                296
10 PATTERN ,pfeld%                                    314
11 LINE (0,0)-(311,239),11,bf                         890
12 [4SPACES]' Ueberschrift                            947
13 COLOR 10,0                                         159
14 LOCATE 2,12:PRINT "Foto-Testpalette"               975
15 ' Testbild zeichnen                                977
16 FOR i = 0 TO 31                                    061
17 [2SPACES]LOCATE 4+i MOD 16,2+INT (i/16)*19         612
18 [2SPACES]COLOR 1,0:PRINT USING "##";i,             243
19 [2SPACES]COLOR i,i:farbe$=SPACE$(16)               407
20 [2SPACES]PRINT farbe$                              192
21 NEXT                                               151
22 [4SPACES]' Menue                                   516
23 WINDOW 3,,(45,175)-( 280,210),16,1                 637
24 COLOR 3,0:CLS:PRINT                                066
25 PRINT " 1-Automatik 2-Einzelbild"                  665
26 check:                                             341
27 q$=INKEY$:IF q$="" THEN check                      855
28  INPUT " Filmdaten: ",Film$                        957
29     ' Automatik                                    919
30 IF q$="1" THEN                                     436
31 COLOR 10,0                                         128
32  FOR j=0 TO 5:FOR i=0 TO 5                         521
33 [2SPACES]CLS:n=n+1:PRINT                           149
34 [2SPACES]PRINT " Filmdaten: ";Film$                699
35 [2SPACES]PRINT " Blende:[4SPACES]";b$(i)           515
36 [2SPACES]PRINT " Zeit:[6SPACES]";z$(j)             882
37 [2SPACES]PRINT TAB(27 );n;                         824
38 [3SPACES]WHILE INKEY$="" :WEND                     286
39  NEXT:NEXT                                         298
40 END IF                                             352
41 [4SPACES]' Einzelbild                              021
42 IF q$="2" THEN                                     291
43  INPUT " Blende:[4SPACES]",Blende$                 425
44  INPUT " Zeit:[6SPACES]",Zeit$                     459
45  COLOR 10,0:CLS:PRINT                              490
46  PRINT " Filmdaten: ";Film$                        752
47  PRINT " Blende:[4SPACES]";Blende$                 454
48  PRINT " Zeit:[6SPACES]";Zeit$                     752
49    WHILE INKEY$="":WEND                            382
50 END IF                                             508
51 beenden:                                           747
52 WINDOW CLOSE 2: WINDOW CLOSE 3                     934
53 SCREEN CLOSE 1                                     017
54 COLOR 1,0:CLS:END                                  854
55 Automatikdaten:                                    971
56 DIM z$(5),b$(5)                                    289
57 FOR i=0 TO 5:READ z$(i):NEXT                       649
58 FOR i=0 TO 5:READ b$(i):NEXT                       337
59 RETURN                                             737
60 DATA 1/30,1 /15,1/8,1/4,1/2,1                      987
61 DATA 2.8,4,5.6,8,11,16                             911
```

Mit diesem Programm bekommt jeder Belichtung und Blende in den Griff.

2000 Mark für das Superlisting

Universelle Konvertierungshilfe für Daten, Texte und Programmfiles

Wenn die Eigenarten mancher Textverarbeitungsprogramme oder Editoren den User zur Verzweifelung treiben, ist „Change" der ersehnte Rettungsring. Jede noch so merkwürdige Codierung läßt sich mit dem „Superlisting" konvertieren. Selbst komplizierte Ersetzungsverfahren sind nun kein Problem mehr.

Wenn Umlaute durch einen Dreier-Code (wie bei WordPerfect) gekennzeichnet sind, Zeilen mit RETURN (bei Editoren), mit SPACE/RETURN (bei Vizawrite) oder ohne Kennung (bei Wordwrapping) enden, werden Bearbeitungen mit Fremdprogrammen zum computerisierten Sysiphus- Schicksal. „Change" bereitet dem Leiden ein Ende. Aufgerufen wird das Programm wie ein CLI-Befehl und sollte sich daher im C-Directory befinden.

Ein weiteres Einsatzgebiet für „Change" ist die automatische Anpassung verschiedener Programmiersprachendialekte. Neue BASIC-Varianten mit abweichenden Befehlssätzen können ebenso leicht bearbeitet werden, wie die kleinen Unterschiede zwischen Aztek und Lattice. Diese Flexibilität wird durch ein Script-File erreicht, in dem die auszuführenden Änderungen enthalten sind.

Die Syntax des Befehls lautet:

```
Change <filename> <scriptname>

<filename>  : Name des zu ändernden Textes
<scriptname>: Name der die Änderungsanweisungen
              enthaltenden Datei
```

Die Leistungsfähigkeit des Programms resultiert aus der besonderen Art und Weise, die Befehle des Script-Files abzuarbeiten. Das Script-File wird zeilenweise interpretiert, wobei jede Zeile eine Änderungsanweisung enthält. Diese hat folgende Struktur:

```
"<alt-String>" = "<neu-String>"

<alt-String>: der zu ersetzende String
<neu-String>: der einzufügende String
```

Beide Strings müssen unter Anführungszeichen gesetzt werden. Nicht druckbare Zeichen innerhalb der Zeichenketten werden durch ein Dollarzeichen, gefolgt von dem hexadezimalen ASCII-Wert, dargestellt. Da das „$"-Zeichen zur Identifizierung von hexadezimalen ASCII- Werten benützt wird, müssen zwei Dollarzeichen zur Darstellung eines normalen „$"-Zeichens eingegeben werden, um eine Verwechslung mit dem Identifizierungszeichen zu vermeiden. Zum Beispiel werden durch

```
"$0a" = " "
```

alle Zeilenvorschübe durch Leerzeichen ersetzt. Soll die Zeichenkette <alt-String> aus dem Text entfernt werden, so ist <neu-String> = "" zu setzen.
Zu Beginn wird die Datei <filename> in den Speicher geladen und die Datei <scriptname> zum Lesen geöffnet. Die Datei <filename> wird im Speicher Zeile für Zeile mit den Änderungsanweisungen aus der Datei <scriptname> behandelt. Es wird also die komplette Datei <filename> mit der ersten, dann mit der zweiten Änderungsanweisung etc. behandelt. Danach wird auf der Diskette die Datei <filename> in <filename>.UNC (UNC = unchanged) umgenannt, und die Datei <filename> aus dem Speicher auf Disk geschrieben.

Folgende Beispiele demonstrieren die Einsatzmöglichkeiten von „Change":

Script-Beispiel 1:

Konvertierung eines Editor-Textes in ein Textverarbeitungsfile

Programmeditoren wie der „Ed" schließen jede Textzeile mit einem Linefeed ab. Beim Einladen in eine Textverarbeitung behindern diese Returns die automatische Formatierung durch das Programm. Ein einfaches Herausnehmen der Returns würde auch Absätze und erwünschte Zeilenabschlüsse (zum Beispiel bei Tabellen und Listingbeispielen) entfernen. Das folgende Script-File beläßt doppelte Linefeeds (entspricht Absatz) und Returns mit folgendem Space (bei Tabellen und Listingbeispielen zu verwenden) im Text.

```
"$0a "     = "?ß-"      ; ersetzt Linefeed+Space
                          durch folgenden Dummy
"$0a$0a" = "?ß+"      ; ersetzt sämtliche doppelte
                          Linefeeds durch Dummy-Code
"$0a"      = " "        ; alle einfachen Linefeeds
                          werden durch Space ersetzt
"?ß+"      = "$0a$0a"   ; Dummy wieder durch doppelte
                          Linefeeds ersetzen
"?ß-"      = "$0a "     ; Dummy durch Linefeed+Space
ersetzen
```

Script-Beispiel 2:

Optische Aufbereitung von Pascal-Programmlistings

Die meisten Programmierer sind zu faul die Key-Words von Programmiersprachen groß zu schreiben. Bei AmigaBASIC erledigt das der Programmeditor, bei anderen Sprachen ist es mit folgendem Script-File möglich, das Programmlisting übersichtlicher zu gestalten.

```
"for"  = "FOR"
"next" = "NEXT"
"if"   = "IF"
"then" = "THEN"
"else" = "ELSE"
...
```

Script-Beispiel 3:

Konvertierung eines WordPerfect-File in ein Textverarbeitungsfile

WordPerfect fügt beim Abspeichern im Filemode vor und nach jedem Sonderzeichen (zum Beispiel Umlauten) das Zeichen $e1 ein. Bei anschließendem Script-File werden diese $e1's aus einem Text entfernt.

```
"$e1ä$e1" = "ä"
"$e1Ä$e1" = "Ä"
"$e1ö$e1" = "ö"
"$e1Ö$e1" = "Ö"
"$e1ü$e1" = "ü"
"$e1Ü$e1" = "Ü"
...
```

Das Hex-Listing von „Change" wird mit Nanomon oder jedem anderen File-Monitor eingegeben. Das Assembler-Listing ist nur aus dokumentarischen Gründen abgedruckt.
(Thomas Meidinger)

Programmname: Change
Sprache: Assembler

Auf Leserdiskette

HEX Eingabe mit Nanomon

```
000 00 00 03 f3 00 00 00 00 00 00 00 02 00 00 00 00 1c
010 00 00 00 01 00 00 01 63 00 00 00 01 00 00 03 e9 1f
020 00 00 01 63 53 40 67 00 04 3c 0c 10 00 20 66 0e ec
030 d1 fc 00 00 00 01 51 c8 ff f2 60 00 04 28 43 f9 8f
040 00 00 04 [illegible] [illegible] 18 0c 01 00 20 67 0a 12 c1 51 c8 4c
050 ff f4 60 00 [illegible] 10 0c 10 00 20 66 0e d1 fc 00 00 2d
060 00 01 51 c8 ff f2 60 00 03 fc 43 f9 00 00 04 70 58
070 12 18 0c 01 00 20 67 10 0c 01 00 0a 67 0a 12 c1 ec
080 51 c8 ff ee 60 00 03 de 43 f9 00 00 04 60 2c 78 e3
090 00 04 4e ae fe 68 23 c0 00 00 04 6c 22 3c 00 01 b9
0a0 00 02 4e ae ff 28 04 80 00 00 27 10 23 c0 00 00 79
0b0 05 74 24 00 22 3c 00 01 00 02 4e ae ff 3a 23 c0 d9
0c0 00 00 05 78 2c 79 00 00 04 6c 22 3c 00 00 04 98 24
0d0 24 3c 00 00 03 ed 4e ae ff e2 23 c0 00 00 04 c4 7d
0e0 67 00 03 60 22 00 42 82 26 3c 00 00 00 01 4e ae 7d
0f0 ff be 22 39 00 00 04 c4 42 82 26 3c ff ff ff ff f1
100 4e ae ff be 23 c0 00 00 04 c8 22 39 00 00 04 c4 d4
110 24 39 00 00 05 78 26 39 00 00 05 74 4e ae ff d6 f5
120 22 39 00 00 04 c4 4e ae ff dc 22 3c 00 00 04 70 db
130 24 3c 00 00 03 ed 4e ae ff e2 23 c0 00 00 04 c0 90
140 67 00 03 00 45 f9 00 00 04 cc 61 00 03 1a 0c 00 b3
150 00 01 66 00 02 6c 0c 39 00 0a 00 00 05 7c 67 ea 00
160 0c 39 00 22 00 00 05 7c 66 e0 61 00 02 fa 0c 00 7e
170 00 01 66 00 02 4c 0c 39 00 24 00 00 05 7c 67 14 8e
180 0c 39 00 22 00 00 05 7c 67 00 00 72 14 f9 00 00 47
190 05 7c 60 d6 61 00 02 d0 0c 00 00 01 66 00 02 22 2f
1a0 0c 39 00 24 00 00 05 7c 66 08 14 fc 00 24 60 00 d3
1b0 ff ba 42 07 10 39 00 00 05 7c 04 00 00 30 0c 00 d4
1c0 00 09 63 08 04 00 00 07 02 00 00 1f e9 08 1e 00 94
1d0 61 00 02 94 0c 00 00 01 66 00 01 e6 10 39 00 00 14
1e0 05 7c 04 00 00 30 0c 00 00 09 63 08 04 00 00 07 1f
1f0 02 00 00 1f de 00 14 c7 60 00 ff 70 61 00 02 68 b0
200 0c 00 00 01 66 00 01 ba 0c 39 00 3d 00 00 05 7c 5b
210 66 ea 14 bc 00 00 95 fc 00 00 04 cc 23 ca 00 00 cd
220 05 6c 45 f9 00 00 05 1c 61 00 02 3c 0c 00 00 01 20
230 66 00 01 8e 0c 39 00 22 00 00 05 7c 66 ea 61 00 23
240 02 26 0c 00 00 01 66 00 01 78 0c 39 00 24 00 00 3f
250 05 7c 67 14 0c 39 00 22 00 00 05 7c 67 00 00 72 28
260 14 f9 00 00 05 7c 60 d6 61 00 01 fc 0c 00 00 01 4b
270 66 00 01 4e 0c 39 00 24 00 00 05 7c 66 08 14 fc 7a
280 00 24 60 00 ff ba 42 07 10 39 00 00 05 7c 04 00 8e
290 00 30 0c 00 00 09 63 08 04 00 00 07 02 00 00 1f 17
2a0 e9 08 1e 00 61 00 01 c0 0c 00 00 01 66 00 01 12 5b
2b0 10 39 00 00 05 7c 04 00 00 30 0c 00 00 09 63 08 37
2c0 04 00 00 07 02 00 00 1f de 00 14 c7 60 00 ff 70 f6
2d0 14 bc 00 00 95 fc 00 00 05 1c 23 ca 00 00 05 70 a7
2e0 20 79 00 00 05 78 43 f9 00 00 04 cc 45 f9 00 00 e4
2f0 05 1c 2e 39 00 00 04 c8 2c 3c 00 00 00 01 1a 10 d8
300 ba 11 67 0a 4a 18 9e 86 6a f4 60 00 fe 38 20 39 ce
310 00 00 05 6c 90 86 1a 31 08 00 ba 30 08 00 66 e4 b2
320 51 c8 ff f4 2a 39 00 00 05 70 ba b9 00 00 05 6c 1d
330 67 00 00 6a 63 00 00 32 28 48 d9 c7 2a 4c db f9 92
340 00 00 05 70 9b f9 00 00 05 6c 1b 24 b1 cc 66 fa a9
350 2a 39 00 00 05 70 9a b9 00 00 05 6c de 85 db b9 db
360 00 00 04 c8 60 00 00 36 28 48 2a 48 d9 f9 00 00 f3
370 05 6c db f9 00 00 05 70 26 48 d7 fc 00 00 00 02 a3
380 d7 c7 1a dc b7 cc 62 fa 2a 39 00 00 05 6c 9a b9 c9
390 00 00 05 70 9e 85 9b b9 00 00 04 c8 20 39 00 00 89
3a0 05 70 67 00 ff 5a 04 80 00 00 00 01 11 b2 08 00 aa
3b0 08 00 51 c8 ff f8 d1 f9 00 00 05 70 60 00 ff 40 d4
3c0 2c 79 00 00 04 6c 22 39 00 00 04 c0 4e ae ff dc 2e
3d0 41 f9 00 00 04 70 43 f9 00 00 04 98 10 d9 4a 11 4e
3e0 66 fa 10 fc 00 2e 10 fc 00 55 10 fc 00 4e 10 fc e2
3f0 00 43 10 bc 00 00 22 3c 00 00 04 70 4e ae ff b8 be
400 24 3c 00 00 04 70 22 3c 00 00 04 98 4e ae ff b2 63
410 22 3c 00 00 04 98 24 3c 00 00 03 ee 4e ae ff e2 01
420 23 c0 00 00 04 c4 22 00 24 39 00 00 05 78 26 39 6c
430 00 00 04 c8 4e ae ff d0 22 39 00 00 04 c4 4e ae 15
440 ff dc 2c 78 00 04 22 79 00 00 05 78 20 39 00 00 04
450 05 74 4e ae ff 2e 22 79 00 00 04 6c 2c 78 00 04 7a
460 4e ae fe 62 4e 75 2c 79 00 00 04 6c 22 39 00 00 41
470 04 c0 24 3c 00 00 05 7c 26 3c 00 00 00 01 4e ae 57
480 ff d6 4e 75 64 6f 73 2e 6c 69 62 72 61 72 79 00 2d
490 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 29
4a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 86
4b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 e3
4c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 40
4d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9d
4e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 fa
4f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 57
500 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9d
510 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 fa
520 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 57
530 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 b4
540 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 11
550 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 6e
560 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 cb
570 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28
580 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 85
590 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 e2
5a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3f
5b0 00 00 03 ec 00 00 00 4c 00 00 00 00 00 00 00 1c a7
```

```
5c0 00 00 00 48 00 00 00 66 00 00 00 74 00 00 00 8a 25
5d0 00 00 00 9c 00 00 00 a2 00 00 00 a8 00 00 00 b8 f4
5e0 00 00 00 d0 00 00 00 e2 00 00 00 e8 00 00 00 ee fb
5f0 00 00 00 f4 00 00 00 fe 00 00 01 08 00 00 01 18 8c
600 00 00 01 22 00 00 01 36 00 00 01 40 00 00 01 56 78
610 00 00 01 60 00 00 01 6a 00 00 01 80 00 00 01 92 e3
620 00 00 01 ba 00 00 01 e8 00 00 01 f4 00 00 01 fa d4
630 00 00 02 00 00 00 02 14 00 00 02 2a 00 00 02 34 27
640 00 00 02 3e 00 00 02 54 00 00 02 66 00 00 02 8e f8
650 00 00 02 b2 00 00 02 b8 00 00 02 be 00 00 02 c4 7b
660 00 00 02 ca 00 00 02 d0 00 00 02 ec 00 00 03 02 e1
670 00 00 03 08 00 00 03 1c 00 00 03 22 00 00 03 2e b1
680 00 00 03 34 00 00 03 3c 00 00 03 4a 00 00 03 50 64
690 00 00 03 66 00 00 03 6c 00 00 03 74 00 00 03 7a 77
6a0 00 00 03 94 00 00 03 9e 00 00 03 a4 00 00 03 ae 78
6b0 00 00 03 b4 00 00 03 d4 00 00 03 de 00 00 03 e4 fb
6c0 00 00 03 ee 00 00 03 fe 00 00 04 06 00 00 04 0c 16
6d0 00 00 04 16 00 00 04 24 00 00 04 2a 00 00 04 34 37
6e0 00 00 04 44 00 00 04 4a 00 00 04 50 00 00 00 00 66
6f0 00 00 03 f2 00 00 03 eb 00 00 00 01 00 00 03 f2 0e
```

Mit „Change" für die Zukunft gerüstet: Alle Daten lassen sich hiermit konvertieren

Auf Leserdiskette

ABC **Eingabe mit beliebigem Editor belegen**
Überprüfen mit Checksum (neue Version)

```
  1 execbase =  4                                     413
  2                                                   413
  3 ; Offsets der Library-Funktionen                  413
  4 OpenLib  = -408                                   004
  5 CloseLib = -414                                   444
  6 Open     = - 30                                   655
  7 Close    = - 36                                   704
  8 Read     = - 42                                   232
  9 Write    = - 48                                   069
 10 AllocMem = -198                                   687
 11 AvailMem = -216                                   282
 12 FreeMem  = -210                                   176
 13 Rename   = - 78                                   025
 14 Seek     = - 66                                   387
 15 Delete   = - 72                                   013
 16                                                   013
 17 start:                                            717
 18 subq      #1,d0        ; Name des Textfiles
    holen                                             972
 19 beq       exit                                    514
 20 search1:                                          107
 21 cmp.b     #" ",(a0)                               384
 22 bne.s     found1                                  959
 23 add.l     #1,a0                                   663
 24 dbf       d0,search1                              014
 25 bra       exit                                    025
 26 found1:                                           772
 27 lea       filename,a1                             861
 28 copy1:                                            467
 29 move.b    (a0)+,d1                                016
 30 cmp.b     #" ",d1                                 788
 31 beq.s     search2                                 413
 32 move.b    d1,(a1)+                                711
 33 dbf       d0,copy1                                375
 34 bra       exit                                    650
 35 search2:                                          512
 36 cmp.b     #" ",(a0)      ; Name des
    Script-Files holen                                653
 37 bne.s     found2                                  701
 38 add.l     #1,a0                                   789
 39 dbf       d0,search2                              173
 40 bra       exit                                    128
 41 found2:                                           156
 42 lea       scriptname,a1                           346
 43 copy2:                                            949
 44 move.b    (a0)+,d1                                378
 45 cmp.b     #" ",d1                                 726
 46 beq.s     getall                                  132
 47 cmp.b     #10,d1                                  145
 48 beq.s     getall                                  463
 49 move.b    d1,(a1)+                                961
 50 dbf       d0,copy2                                082
 51 bra       exit                                    221
 52 getall:                                           124
 53 lea       dosname,a1       ; DOS-Library
    öffnen                                            099
 54 move.l    execbase.w,a6                           724
 55 jsr       OpenLib(a6)                             004
 56 move.l    d0,dosbase                              598
 57 move.l    #$10002,d1       ; Arbeitsspeicher
    anfordern                                         706
 58 jsr       AvailMem(a6)                            969
 59 sub.l     #10000,d0                               037
 60 move.l    d0,memlen                               468
 61 move.l    d0,d2                                   711
 62 move.l    #$10002,d1                              327
 63 jsr       AllocMem(a6)                            580
 64 move.l    d0,memstart                             198
 65 move.l    dosbase,a6                              779
 66 move.l    #filename,d1     ; Textfile öffnen
    und einlesen                                      077
 67 move.l    #1005,d2                                109
 68 jsr       Open(a6)                                790
 69 move.l    d0,filehandle                           079
 70 beq       exit1                                   178
 71 move.l    d0,d1                                   068
 72 clr.l     d2                                      433
 73 move.l    #1,d3                                   113
 74 jsr       Seek(a6)                                736
 75 move.l    filehandle,d1                           738
 76 clr.l     d2                                      015
 77 move.l    #-1,d3                                  968
 78 jsr       Seek(a6)                                263
 79 move.l    d0,filelen                              015
 80 move.l    filehandle,d1                           857
 81 move.l    memstart,d2                             377
 82 move.l    memlen,d3                               483
 83 jsr       Read(a6)                                714
 84 move.l    filehandle,d1                           772
 85 jsr       Close(a6)                               985
 86 move.l    #scriptname,d1   ; Scriptfile
    öffnen                                            412
 87 move.l    #1005,d2                                760
 88 jsr       Open(a6)                                049
 89 move.l    d0,scripthandle                         411
 90 beq       exit1                                   246
 91                                                   246
 92 getline:                                          724
 93 lea       searchmem,a2     ; eine Zeile des
    Scripts einlesen                                  750
 94 getbyte:                   ; und umsetzen         304
 95 bsr       bytelesen                               910
 96 cmp.b     #1,d0                                   634
 97 bne       endofchange                             313
 98 cmp.b     #10,dummymem                            920
 99 beq.s     getbyte                                 325
100 cmp.b     #34,dummymem                            202
101 bne.s     getbyte                                 264
102 getscript:                                        403
103 bsr       bytelesen                               077
104 cmp.b     #1,d0                                   773
105 bne       endofchange                             032
106 cmp.b     #"$",dummymem                           042
107 beq.s     sonderzeichen                           208
108 cmp.b     #34,dummymem                            077
109 beq       suchgleich                              180
110 move.b    dummymem,(a2)+                          765
111 bra.s     getscript                               996
112 sonderzeichen:                                    859
113 bsr       bytelesen                               757
114 cmp.b     #1,d0                                   997
115 bne       endofchange                             104
116 cmp.b     #"$",dummymem                           994
117 bne.s     hexwert                                 951
118 move.b    #"$",(a2)+                              308
119 bra       getscript                               318
120 hexwert:                                          323
121 clr.b     d7                                      171
122 move.b    dummymem,d0                             513
123 sub.b     #$30,d0                                 925
124 cmp.b     #$09,d0                                 409
125 bls.s     hexwert1                                311
126 sub.b     #$07,d0                                 835
127 and.b     #$1f,d0                                 720
128 hexwert1:                                         182
129 lsl.b     #4,d0                                   796
130 move.b    d0,d7                                   602
131 bsr       bytelesen                               040
132 cmp.b     #1,d0                                   116
133 bne       endofchange                             819
134 move.b    dummymem,d0                             993
135 sub.b     #$30,d0                                 485
136 cmp.b     #$09,d0                                 681
137 bls.s     hexwert2                                944
138 sub.b     #$07,d0                                 472
139 and.b     #$1f,d0                                 249
140 hexwert2:                                         888
141 add.b     d0,d7                                   764
142 move.b    d7,(a2)+                                649
143 bra       getscript                               831
144                                                   831
145 suchgleich:                                       012
146 bsr       bytelesen                               738
147 cmp.b     #1,d0                                   734
148 bne       endofchange                             813
149 cmp.b     #"=",dummymem                           900
150 bne.s     suchgleich                              885
151                                                   885
```

```
152 move.b  #0,(a2)                          123
153 sub.l   #searchmem,a2                    878
154 move.l  a2,searchlen                     439
155 lea     changemem,a2                     817
156                                          817
157 getbytel:                                284
158 bsr     bytelesen                        970
159 cmp.b   #1,d0                            742
160 bne     endofchange                      853
161 cmp.b   #34,dummymem                     346
162 bne.s   getbytel                         961
163 getchange:                               401
164 bsr     bytelesen                        107
165 cmp.b   #1,d0                            827
166 bne     endofchange                      254
167 cmp.b   #"$",dummymem                    712
168 beq.s   sonderzeichen2                   984
169 cmp.b   #34,dummymem                     661
170 beq     haveall                          306
171 move.b  dummymem,(a2)+                   555
172 bra.s   getchange                        179
173 sonderzeichen2:                          580
174 bsr     bytelesen                        082
175 cmp.b   #1,d0                            782
176 bne     endofchange                      053
177 cmp.b   #"$",dummymem                    651
178 bne.s   hexwert3                         703
179 move.b  #"$",(a2)+                       644
180 bra     getchange                        543
181 hexwert3:                                051
182 clr.b   d7                               435
183 move.b  dummymem,d0                      585
184 sub.b   #$30,d0                          709
185 cmp.b   #$09,d0                          009
186 bls.s   hexwert4                         530
187 sub.b   #$07,d0                          690
188 and.b   #$1f,d0                          619
189 hexwert4:                                900
190 lsl.b   #4,d0                            450
191 move.b  d0,d7                            288
192 bsr     bytelesen                        958
193 cmp.b   #1,d0                            130
194 bne     endofchange                      889
195 move.b  dummymem,d0                      895
196 sub.b   #$30,d0                          803
197 cmp.b   #$09,d0                          751
198 bls.s   hexwert5                         289
199 sub.b   #$07,d0                          925
200 and.b   #$1f,d0                          426
201 hexwert5:                                036
202 add.b   d0,d7                            688
203 move.b  d7,(a2)+                         269
204 bra     getchange                        156
205                                          156
206 haveall:                                 455
207 move.b  #0,(a2)                          485
208 sub.l   #changemem,a2                    400
209 move.l  a2,changelen                     281
210                                          281
211 move.l  memstart,a0      ; eine Scriptzeile
     am Textfile ausführen                   208
212 lea     searchmem,a1                     385
213 lea     changemem,a2                     355
214 move.l  filelen,d7                       414
215 move.l  #1,d6                            081
216                                          081
217 suche:                                   311
218 move.b  (a0),d5                          945
219 cmp.b   (a1),d5                          361
220 beq.s   firstsame                        216
221 contsuche:                               974
222 tst.b   (a0)+                            546
223 sub.l   d6,d7                            347
224 bpl.s   suche                            930
225 bra     getline                          163
226                                          163
227 firstsame:                               427
228 move.l  searchlen,d0                     793
229 sub.l   d6,d0                            375
230 compare:                                 184
231 move.b  0(a1,d0),d5                      347
232 cmp.b   0(a0,d0),d5                      278
233 bne.s   contsuche                        752
234 dbf     d0,compare                       383
235 move.l  changelen,d5     ; alten Text durch
     neuen austauschen                       458
236 cmp.l   searchlen,d5                     694
237 beq     einsetzten                       591
238 bls     makeshorter                      413
239 move.l  a0,a4                            768
240 add.l   d7,a4                            327
241 move.l  a4,a5                            991
242 add.l   changelen,a5                     917
243 sub.l   searchlen,a5                     252
244 makelonger:                              835
245 move.b  -(a4),-(a5)                      373
246 cmp.l   a4,a0                            573
247 bne.s   makelonger                       324
248 move.l  changelen,d5                     323
249 sub.l   searchlen,d5                     129
250 add.l   d5,d7                            788
251 add.l   d5,filelen                       331
```

```
252 bra     einsetzten                       585
253 makeshorter:                             808
254 move.l  a0,a4                            771
255 move.l  a0,a5                            135
256 add.l   searchlen,a4                     396
257 add.l   changelen,a5                     034
258 move.l  a0,a3                            500
259 add.l   #2,a3                            808
260 add.l   d7,a3                            398
261 makeshorter2:                            267
262 move.b  (a4)+,(a5)+                      789
263 cmp.l   a4,a3                            096
264 bhi.s   makeshorter2                     211
265 move.l  searchlen,d5                     846
266 sub.l   changelen,d5                     348
267 sub.l   d5,d7                            432
268 sub.l   d5,filelen                       712
269 einsetzten:                              139
270 move.l  changelen,d0                     617
271 beq     suche                            533
272 sub.l   #1,d0                            233
273 copiere:                                 734
274 move.b  0(a2,d0),0(a0,d0)                163
275 dbf     d0,copiere                       638
276 add.l   changelen,a0                     047
277 bra     suche                            884
278                                          884
279 endofchange:                             600
280 move.l  dosbase,a6        ; Scriptfile
     schließen                               113

281 move.l  scripthandle,d1                  344
282 jsr     Close(a6)                        221
283 lea     scriptname,a0     ; original
     Textfile umbenennen in <name>.UNC       430
284 lea     filename,a1                      199
285 copyname:                                025
286 move.b  (a1)+,(a0)+                      779
287 tst.b   (a1)                             677
288 bne.s   copyname                         823
289 move.b  #".",(a0)+                       852
290 move.b  #"U",(a0)+                       856
291 move.b  #"N",(a0)+                       225
292 move.b  #"C",(a0)+                       191
293 move.b  #0,(a0)                          515
294 move.l  #scriptname,d1                   974
295 jsr     Delete(a6)                       192
296 move.l  #scriptname,d2                   672
297 move.l  #filename,d1                     570
298 jsr     Rename(a6)                       285
299 move.l  #filename,d1      ; geändertes
     Textfile abspeichern                    767
300 move.l  #1006,d2                         612
301 jsr     Open(a6)                         501
302 move.l  d0,filehandle                    326
303 move.l  d0,d1                            664
304 move.l  memstart,d2                      976
305 move.l  filelen,d3                       891
306 jsr     Write(a6)                        389
307 move.l  filehandle,d1                    663
308 jsr     Close(a6)                        824
309                                          824
310 exitl:                                   649
311 move.l  execbase.w,a6     ; Arbeitsspeicher
     freigeben                               074
312 move.l  memstart,a1                      658
313 move.l  memlen,d0                        289
314 jsr     FreeMem(a6)                      193
315 move.l  dosbase,a1                       621
316 move.l  execbase.w,a6                    766
317 jsr     CloseLib(a6)                     042
318 exit:                                    190
319 rts                                      047
320                                          047
321 ; Unterroutinen                          047
322 bytelesen:                               856
323 move.l  dosbase,a6        ; ein Byte vom
     Scriptfile einlesen                     369
324 move.l  scripthandle,d1                  600
325 move.l  #dummymem,d2                     149
326 move.l  #1,d3                            021
327 jsr     Read(a6)                         372
328 rts                                      541
329                                          541
330 dosname: dc.b "dos.library",0            830
331 even                                     628
332 dosbase: dc.l 0                          696
333 scriptname: blk.b 40,0                   677
334 filename:   blk.b 40,0                   777
335 scripthandle: dc.l 0                     265
336 filehandle:   dc.l 0                     176
337 filelen:      dc.l 0                     198
338                                          198
339 searchmem: blk.b 80,0                    470
340 changemem: blk.b 80,0                    214
341 searchlen: dc.l 0                        494
342 changelen: dc.l 0                        854
343 memlen:    dc.l 0                        139
344 memstart:  dc.l 0                        979
345 dummymem:  blk.b 10,0                    626
346                                          626
347                                          818
```

Dokumentation des „Change"-Assemblers

Automatische Fehlersuche mit Memwatch

Wer programmiert macht auch Fehler. Diese zu beheben, kann den Hauptteil der Entwicklungszeit eines größeren Softwareprojekts ausmachen. Aber warum sich nicht vom Computer helfen lassen? Memwatch zeigt, wie sich die Fehlersuche automatisieren läßt.

Auf multitaskingfähigen Rechnern ist es wichtig, daß jedes Programm alle verwendeten Systemressourcen spätestens beim Programmschluß wieder freigibt. Leider werden gerade hier leicht Fehler gemacht, deren Korrektur sehr schwierig ist. Dabei ist es weniger das Problem, die verantwortliche Ressource freizugeben. Sie überhaupt zu finden, macht die größte Mühe. Ein vergessener CloseWindow()-Befehl wird deutlich, wenn ein Fenster noch offensteht, das zugehörige Programm aber längst nicht mehr aktuell ist. Nicht freigegebene FileLocks bewirken, daß man keine Daten mehr in eine Datei schreiben oder ein Verzeichnis nicht löschen kann, denn AmigaDOS glaubt ja verständlicherweise, daß das vergessene FileLock noch benötigt wird. 'Error code 202: object in use' ist dann die Folge. Ein Blick auf die Speicheranzeige der Workbench schafft Klarheit: Wenn die freie Speichermenge vor dem Start eines Programms nicht mit der nach dem Programmende übereinstimmt, hat entweder das Programm einen Fehler bei der Speicherbelegung gemacht, oder man ist einem Systemfehler auf der Spur, was allerdings nicht sehr wahrscheinlich ist. War das Aufspüren offener Fenster noch einfach, so wird es umso schwieriger bei FileLocks. Deshalb greift Memwatch an der Wurzel des Problems an: Beim Anfordern des nötigen Speicherplatzes mit dem AllocMem()-Befehl.
Die Anwendung des Programms ist sehr einfach: Man fügt einfach am Anfang des verdächtigen C-Programms die Zeilen

```
if MEMWATCH
 struct memwatch *mainwatch;
#endif
```

als globale Definition ein. Um die Diagnose von Memwatch zu erhalten, wenn das Programmende erreicht ist, muß vor dem betreffenden return- oder Exit()-Befehl

```
#if MEMWATCH
 reportmw();
#endif
```

stehen. Diese Methode wird im Beispielprogramm memwaste.c nochmals verdeutlicht. Da MEMWATCH dem Compiler noch nicht bekannt ist, hat sich nichts an der Funktion des Programms geändert. Um die Überwachung des Speichers zu starten, werden alle AllocMem()- und FreeMem()-Befehle durch die Routinen allocmw() bzw. freemw() ersetzt, was durch das Präprozessorkommando define geschieht. Jetzt muß das Symbol MEMWATCH definiert werden. Damit man nicht in jedes Sourcefile die entsprechenden Befehle eingeben muß, sind alle notwendigen Operationen im Headerfile memwatch.h zusammengefaßt. Nun kommt der langweilige Teil der Geschichte: Jedes zum Programm gehörige Sourcefile, in dem Speicher über AllocMem() und FreeMem() angefordert oder freigegeben wird, muß neu compiliert werden, nachdem memwatch.h (meist über ein globales Include-File) eingebunden wurde. Das fertige Objekt-Modul wird dann mit memwatch.o (dem fertig compilierten memwatch.c) zusammengelinkt werden, und fertig ist die Speicherüberwachung.

allocmw() legt eine Liste der belegten Speichersegmente an. Aus dieser Liste werden alle korrekt freigegebenen Speicherblöcke von freemw() wieder entfernt. Beim Programmschluß wird die Funktion reportmw() aufgerufen, die die entstandene Liste überprüft. Wurden irgendwann belegte Speichersegmente gar nicht, fehlerhaft (oder nicht mit freemw()!) freigegeben, werden sie jetzt mit Adresse, Blockgröße und Inhalt aufgelistet. Zum Schluß wird die Summe der nicht freigegebenen Bytes ausgegeben. Natürlich kann die Routine reportmw() auch am Ende eines wichtigen Programmteils benutzt werden. Da sie die gesamte Memwatch-Liste freigibt, wird danach wieder bei Null angefangen. Die Fehlermeldungen werden mit printf() ausgegeben! Falls das Programm aussteigt, wenn es von Workbench aus gestartet wird, ist ein Startversuch über das CLI ratsam. Ebenso ließe sich die printf()-Anweisung durch eine persönliche Ausgabemethode ersetzen. Beim Ablauf des Programms werden Fehlermeldungen ausgegeben, sobald das Programm über freemw() Speicher freigibt, den es nicht per allocmw() belegt hat („Unknown pointer freed"). In den meisten Fällen besagt dies, daß nicht alle Dateien recompiliert wurden. Die Meldung „Block size mismatch" bedeutet, daß die Größenangabe bei der Freigabe eines Speicherblocks nicht mit der bei seiner Anforderung übereinstimmt.

Jetzt weiß der verzweifelte User, was nicht freigegeben wurde. Es bleibt herauszufinden, was die Daten bedeuten und wo sie belegt wurden. Oft reicht die ASCII-Darstellung bei der Ausgabe schon, um Textdaten zu identifizieren. Ist das nicht der Fall, ist die Angabe der Blockgröße das wichtigste Hilfsmittel, denn Speicherplatzanforderungen über AllocMem() werden meistens für Strings oder Strukturen gemacht. Strukturen haben immer eine feste Größe; wenn reportmw() also regelmäßig ein Speichersegment der gleichen Größe ausgibt, haben Sie wahrscheinlich vergessen, eine Struktur freizugeben. Um den Typ einer Struktur bekannter Größe herauszufinden, kann man diese einfach mit den Bytezahlen aller im Programm benutzten Strukturen vergleichen; ein dazu benötigtes Programm, das die Strukturgrößen bestimmt (sizeof-Kommando), ist leicht geschrieben. Noch praktischer sind Debugging-Routinen, die sich an bestimmten Stellen des Programms (zum Beispiel printf()-Ausgaben) melden. Hier braucht lediglich dieser Modus aktiviert werden und in memwatch.c die Zeile

```
if(size==GESUCHTE_STRUKTURGRÖSSE)
      printf("HIER!\n");
```

in die allocmw()-Routine eingefügt werden. Es ist so leicht möglich, die gesuchten AllocMem()-Befehle zu lokalisieren. Man kann die Memwatch-Methode natürlich auch für andere Systemressourcen verwenden, indem man z.B. die Befehle Lock() und Unlock() durch entsprechende mw()-Funktionen ersetzt und die Routinen von memwatch.c umschreibt. Auch die Kontrolle mehrerer Ressourcen gleichzeitig ist lediglich eine Fleißaufgabe, indem man mainwatch zu einem Zeiger auf eine

```
struct mainwatch
{
 struct memwatch *memmain;
 struct resource_1 *res_1;
 . . . struct resource_n *res_n;
```

macht und die einzelnen Listen an dieser Stelle beginnen läßt, statt jeden Listenanfang einzeln zu übergeben. So wird auch der zusätzliche Aufwand beim Austesten von ROM-fähigem Code gemindert. Ein weiterer Vorteil an Memwatch ist, daß das zugrundeliegende Prinzip in jede Computersprache übertragen werden kann, indem man statt einem define-Befehl einfach den Sourcecode direkt verändert. Aus naheliegenden Gründen wurde C als Programmiersprache gewählt. Das Beispielprogramm kompiliert, linkt und läuft unter Lattice C 4.0 „fehlerfrei".

(Achim Albert)

Programmname: Memwatch
Sprache: C

Auf Leserdiskette

ABC Eingabe mit beliebigem Editor (ohne Zeilennummern)
Überprüfen mit checksum (alte Version)

```
 1 /*                                   *        653
 2  *  memwatch.c - main code segment   *        653
 3  *                                   *        653
 4  *       20-May-1988 Achim Albert    *        653
 5  *                                   */       358
 6                                               358
 7 #include<exec/types.h>                        010
 8 #include<exec/memory.h>                       470
 9                                               470
10 #include<ctype.h> /* to define isprint() */   908
11 #include<stdio.h>                             798
12                                               798
13 /* only for Lattice C 4.0 */                  892
14                                               892
15 #include<proto/exec.h>                        591
16 #include<proto/dos.h>                         663
17                                               663
18                                               663
19 #define BYTES_PER_LINE 10                     286
20                                               286
21 struct memwatch                               397
22 {                                             188
23          struct memwatch *nextmem;            982
24          UBYTE *pointer;                      169
25          LONG size;                           559
26 };                                            947
27                                               947
28 STATIC struct memwatch *endmw(struct memwatch *);  576
29                                               576
30 STATIC struct memwatch *endmw(start)          583
31  struct memwatch *start;                      485
32 {                                             284
33  struct memwatch *seeker=start;               294
34                                               294
35  while(seeker!=NULL)                          280
36  {                                            715
37   start=seeker;                               344
38   seeker=seeker->nextmem;                     703
39  }                                            072
40  return(start);                               750
41 }                                             683
42                                               683
43 UBYTE *allocmw(size,type)                     299
44  LONG size,type;                              759
45 {                                             798
46  extern struct memwatch *mainwatch;           204
47                                               204
48  struct memwatch *newmw;                      478
49  UBYTE *watch=AllocMem(size,type);            616
50                                               616
51  if(mainwatch==NULL)                          069
52  {                                            020
53   if(!(newmw=mainwatch=(struct memwatch *)    502
54    AllocMem(sizeof(struct
     memwatch),MEMF_CLEAR|MEMF_PUBLIC)))         933
55   {                                           012
56    printf("help");                            960
57    Exit(21);                                  877
58   }                                           286
59  }                                            795
60  else                                         948
61  {                                            159
62   newmw=endmw(mainwatch);                     089
63   if(!(newmw=newmw->nextmem=(struct memwatch *)  690
64    AllocMem(sizeof(struct
     memwatch),MEMF_CLEAR|MEMF_PUBLIC)))         065
65   {                                           968
66    printf("help");                            644
67    Exit(22);                                  650
68   }                                           383
69  }                                            008
70  newmw->size=size;                            499
71  newmw->pointer=watch;                        836
72  return(watch);                               931
73 }                                             988
74                                               988
75 VOID freemw(pointer,size)                     697
76  UBYTE *pointer;                              816
77  LONG size;                                   302
78 {                                             001
79  extern struct memwatch *mainwatch;           039
80                                               039
81  struct memwatch *seeker,*lastseek=NULL;      952
82                                               952
83  if(mainwatch!=NULL)                          921
84  {                                            856
85   seeker=mainwatch;                           503
86   while(seeker->pointer!=pointer)             871
87   {                                           206
88    if(seeker->nextmem!=NULL)                  847
89    {                                          894
90     lastseek=seeker;                          865
91     seeker=seeker->nextmem;                   676
92    }                                          721
```

```
 93     else                                                   970
 94     {                                                      445
 95      printf("Unknown pointer freeá: $%lx,
          %ld\n",pointer,size);                                373
 96      return;                                               564
 97     }                                                      313
 98    }                                                       098
 99    if(seeker->size!=size)                                  560
100    {                                                       259
101     printf("Block size mismatch: $%lx, %ld !=
         %ld\n",pointer,size,seeker-                           455
102 >size);                                                    912
103     return;                                                987
104    }                                                       668
105    if(lastseek==NULL)                                      781
106    {                                                       036
107     lastseek=mainwatch;                                    204
108     mainwatch=mainwatch->nextmem;                          549
109     FreeMem((char *)lastseek,sizeof(struct
         memwatch));                                           355
110    }                                                       644
111    else                                                    277
112    {                                                       676
113     lastseek->nextmem=seeker->nextmem;                     249
114     FreeMem((char *)seeker,sizeof(struct memwatch));       442
115    }                                                       775
116   }                                                        008
117   else                                                     241
118   {                                                        208
119    printf("Unknown pointer freed: $%lx,
        %ld\n",pointer,size);                                  672
120    return;                                                 275
121   }                                                        652
122   FreeMem(pointer,size);                                   241
123 }                                                          210
124                                                            210
125 void reportmw()                                            005
126 {                                                          188
127  extern struct memwatch *mainwatch;                        306
128                                                            306
129  struct memwatch *seeker=mainwatch,*lastseek;              268
130  LONG sum=0;                                               321
131  LONG i,j;                                                 499
132                                                            499
133  if(mainwatch==NULL)                                       492
134  {                                                         375
135   printf("MemWatch: No memory left!\n");                   110
136   return;                                                  369
137  }                                                         826
138  while(seeker!=NULL)                                       932
139  {                                                         999
140   lastseek=seeker;                                         522
141   printf("pointer $%lx, size %ld
       left:\n",seeker->pointer,seeker->size);                 090
142   for(i=0;i<seeker->size;i+=BYTES_PER_LINE)                717
143   {                                                        252
144    printf("%3ld:",i);                                      138
145    for(j=0;j<BYTES_PER_LINE;j++)                           333
146     if(i+j<seeker->size)                                   740
147      printf("%3ld,",*(seeker->pointer+i+j));               030
148     else                                                   847
149      printf("[4SPACES]");                                  858
150    printf("[4SPACES]");                                    937
151    for(j=0;j<BYTES_PER_LINE;j++)                           712
152     if(isprint(*(seeker->pointer+i+j))&&i+j<seeker->
        size)                                                  300
153      printf("%c",*(seeker->pointer+i+j));                  166
154     else                                                   087
155      printf(" ");                                          698
156    printf("\n");                                           531
157   }                                                        884
158   sum+=seeker->size;                                       245
159   seeker=seeker->nextmem;                                  776
160   FreeMem((char *)lastseek,sizeof(struct memwatch));       918
161  }                                                         819
162  printf("sum=%ld ($%lx)\n",sum,sum);                       459
163  mainwatch=NULL;                                           210
164 }                                                          807
165                                                            807
```

C-Source von Memwatch

```
 1 /*
      *                                                        653
 2 *  memwatch.h - include this header file when using
                   *                                           653
 3 *               memwatch debugging routines
                     *                                         653
 4 *
      *                                                        653
 5 *                  20-May-1988 Achim Albert
                     *                                         653
 6 *
      */                                                       358
 7                                                             358
 8 #include<exec/types.h>                                      010
 9                                                             010
10 #define AllocMem allocmw                                    117
11 #define FreeMem freemw                                      014
12                                                             014
13 #define MEMWATCH 1                                          427
14                                                             427
15 void reportmw(void);                                        835
16 char *allocmw(long,long);                                   106
17 void freemw(char *,long);                                   090
18                                                             090
19 struct memwatch                                             417
20 {                                                           448
21          struct memwatch *nextmem;                          114
22          UBYTE *pointer;                                    973
23          LONG size;                                         347
24 };                                                          935
```

Dieser Header muß in das zu testende Programm eingebunden werden.

```
 1 /*                                                    *   653
 2  *   memwaste.c - demonstration for memwatch utility  *   653
 3  *                                                    *   653
 4  *              20-May-1988 Achim Albert              *   653
 5  *                                                    */  358
 6                                                            358
 7 /* link with                                          *   763
 8  *    blink from lib:c.o+memwaste.o+memwatch.o        *   763
 9  *          to ram:memwaste                           *   763
10  *          lib lib:lc.lib+lib:amiga.lib              */  516
11                                                            516
12 #include<exec/types.h>                                     440
13 #include<exec/memory.h>                                    788
14 #include<string.h>                                         486
15                                                            486
16 /* Lattice C 4.0 users only!                   *           867
17  * (you also needn't link with amiga.lib) */               708
18                                                            708
19 #include<proto/exec.h>                                     831
20                                                            831
21 char *allocwithout(long,long);                             194
22                                                            194
23                                                            194
24 char *allocwithout(size,type)                              771
25  long size,type;                                           943
26 {                                                          094
27  return(AllocMem(size,type));                              914
28 }                                                          767
29                                                            767
30                                                            767
31 #include"memwatch.h"                                       930
32                                                            930
33 #if MEMWATCH                                               590
34 struct memwatch *mainwatch;                                018
35 #endif                                                     971
36                                                            971
37 void main()                                                603
38 {                                                          818
39  char *x,*y,*z;                                            908
40                                                            908
41  x=AllocMem(100,MEMF_CLEAR);                               367
42  y=AllocMem(120,MEMF_CLEAR);                               465
43  z=allocwithout(100,MEMF_CLEAR);                           007
44                                                            007
45  strcpy(x,"Dieser Block wird gar nicht freigegeben !");    297
46  strcpy(y,"Und hier werden 20 Bytes stehengelassen !");    094
47  strcpy(z,"Hier hat jemand vergessen, mit memwatch zu
     kompilieren !");                                         442
48                                                            442
49  FreeMem(y,100);                                           545
50  FreeMem(z,100);                                           709
51  #if MEMWATCH                                              105
52   reportmw();                                              017
53  #endif                                                    250
54 }                                                          327
55                                                            327
```

Beispiel eines Programms, das getestet werden soll.

```
1>ram:memwaste

Block size mismatch: $c0a338, 100 != 120
Unknown pointer freed: $c0e358, 100
pointer $c08988, size 100 left:
  0: 68,105,101,115,101,114, 32, 66,108,111,    Dieser Blo
 10: 99,107, 32,119,105,114,100, 32,103, 97,    ck wird ga
 20:114, 32,110,105, 99,104,116, 32,102,114,    r nicht fr
 30:101,105,103,101,103,101, 98,101,110, 32,    eigegeben
 40: 33,  0,  0,  0,  0,  0,  0,  0,  0,  0,    !
 50:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 60:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 70:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 80:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 90:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
pointer $c0a338, size 120 left:
  0: 85,110,100, 32,104,105,101,114, 32,119,    Und hier w
 10:101,114,100,101,110, 32, 50, 48, 32, 66,    erden 20 B
 20:121,116,101,115, 32,115,116,101,104,101,    ytes stehe
 30:110,103,101,108, 97,115,115,101,110, 32,    ngelassen
 40: 33,  0,  0,  0,  0,  0,  0,  0,  0,  0,    !
 50:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 60:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 70:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 80:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 90:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
100:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
110:  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
sum=220 ($dc)
```

So sieht eine typische Fehlermeldung von Memwatch aus.

Verbotene Waffe

Mit „Silent Service“ ist eine detaillierte Simulation amerikanischer U-Boot-Einsätze auf den Markt gekommen. Der Spieler versetzt sich in die Rolle des Kapitäns und wird mit den gleichen Informationen, Problemen und Möglichkeiten konfrontiert, wie sie ein wirklicher U-Boot-Kapitän im Zweiten Weltkrieg hatte. Grafik- und Toneffekte ergeben einen so hohen Grad von Realität, daß das pazifische Kriegsspiel am 30. Juli 88 endgültig von der Bundesprüfstelle aus dem Verkehr gezogen wurde.

Microprose ist nach eigenen Angaben ständig bestrebt, realistische Simulationen von hoher Qualität anzubieten, die Aufregung schaffen, eine Herausforderung darstellen und gleichzeitig Wissen vermitteln. Für die U-Boot-Simulation „Silent Service“ (deutsch: Das U-Boot) trifft dies durchaus zu. Umstritten bleibt freilich die Frage nach den pädagogischen Wirkungen einer so intensiven Beschäftigung mit kriegerischen Ereignissen der Jahre 1942 — 1944. Zugegeben, der U-Boot-Kampf hat in seiner Dimension des Schreckens eine eigene Faszination und ist einzigartig in der Art, wie er eine gründliche Planung, blitzschnelles Handeln, Geschick und Einfallsreichtum verlangt. Sid Meier, der im Auftrag von Microprose-Boss Bill Stealey die Simulation entwarf und entwickelte, war von Anfang an überzeugt, daß dieses Thema besonders geeignet für eine Computer-Simulation sei: „Unser vorrangiges Ziel bestand darin, eine Simulation von größerer Detailtreue, von größerem Realismus und größerer Vielseitigkeit zu schaffen als andere Simulationen, ohne dabei die Spielbarkeit zu beeinträchtigen.“

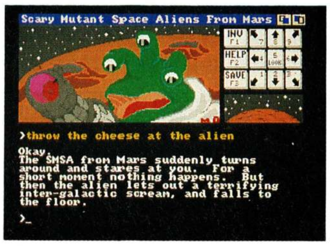
Das Team von Microprose hat es mit Hilfe einiger ehemaliger U-Boot-Offiziere geschafft, eine fast grenzenlose Vielfalt an Situationen und Möglichkeiten der Wahl der gewünschten Einsatzrouten und Patrouillengebiete zu bieten. Als Hindernis bei der Entwicklung erwies sich der Zeitfaktor; in der Realität konnten U-Boot-Einsätze viele Stunden dauern, manchmal Tage, in denen der Kapitän sein U-Boot in eine günstige Angriffsposition manövrierte und seine Feinde auf Zick-Zack-Kurs gingen, um ihn zu verwirren. Wenn das Gefecht jedoch begann, wurden den Torpedoangriffe in Minuten und Sekunden gemessen, denn ein gezielter Wasserbombenangriff konnte ein U-Boot mit einer einzigen Explosion zerdrücken. Als Lösungsmöglichkeit hat sich das Programmiererteam überlegt, Sichtweiten, Bewegungsmaßstäbe, Drehgeschwindigkeiten und so weiter zu ändern und eine sogenannte „Badewannen“-Simulation mit ständigen Wasserbombenangriffen und exorbitanten Manövern zu produzieren. Doch ihrer Auffassung nach wäre ein Großteil der Taktik und des Geschicks, das ein richtiger U-Boot-Kapitän besitzen mußte, vernachlässigt und das ursprüngliche Entwicklungsziel dadurch nicht erreicht worden. Deshalb implementierte man ein Zeitmaßsystem, das es dem Spieler ermöglicht, Geschwindigkeit und Simulation bei längeren Phasen zu beschleunigen. Der Computer beobachtet ständig sämtliche Schiffe, Torpedos und natürlich das U-Boot. Diese Informationen werden sodann gefiltert, um dem Spieler den Standpunkt des U-Boot-Kommandanten zu vermitteln, das heißt, Informationen, die einem realen U-Boot-Kommandanten nicht zugänglich wären, bleiben auch dem Spieler verborgen. Als besonders interessant empfanden die Mitarbeiter von Microprose die Gelegenheit, reale U-Boot-Taktiken zu erlernen und anzuwenden. Es stellte sich heraus, daß bei der ungeheuren Vielfalt an taktischen Problemen Routinelösungen nicht zur Verfügung stehen. Jede Situation muß auf Grund einer Analyse der identischen Faktoren, wie sie sich in der Vergangenheit auf reale U-Boot-Gefechte ausgewirkt haben, bewältigt werden. Daß dieses Spiel letztlich indiziert wurde, erklärt sich aus einer Passage der Spielanleitung: „Wir hoffen, daß auch Sie diese Simulation als mehr empfinden werden als nur ein künstlich erzeugtes Spiel. Wenn Sie auch nur ein wenig zusammenzucken, wenn über Ihnen die Wasserbomben ins Meer rollen, wenn Sie auch nur einen Schimmer von Befriedigung spüren, wenn Ihre Torpedos ihr Ziel erreichen . . ., dann war unsere Mühe nicht umsonst.“ Bei all ihrer Liebe zum Detail haben die Schöpfer von „Silent Service“ offensichtlich übersehen, daß es gerade die von ihnen erzeugte Wirklichkeitsnähe ist, die es verbieten sollte, zu einem Geschehen, das über 50 Millionen Tote zur Folge hatte, „viel Glück und fröhliche Jagd“ zu wünschen.
(mss)



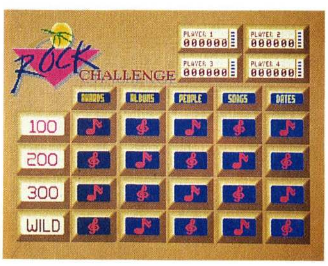
Der Kapitän in der Kommandozentrale

Kurz notiert:

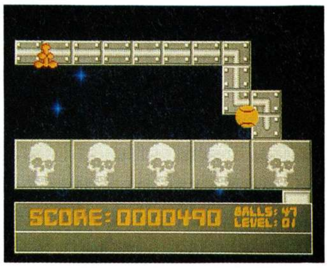
Der amerikanische Softwarehersteller ReadySoft stellt vor: **„Scary Mutant Space Aliens from Mars“.** Hinter diesem atemberaubenden Titel verbirgt sich ein Textadventure mit digitalisierten Grafiken und Sounds. Spieler können hier die eigenartigsten Aliens kennenlernen.



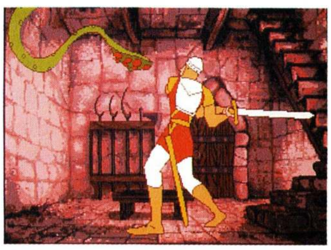
„Rock Challenge“ hat nichts mit Steinesammeln, sondern mit Rockmusik zu tun. Es handelt sich hierbei um einen regelrechten Wettbewerb: Wer weiß am meisten über diese Musikrichtung? Fragen zur Rockszene der Fünfziger Jahre bis heute fordern den Rhythmusfreak. Wer sich in diesem Thema nicht auskennt, kann sein einschlägiges Wissen mit diesem Programm erweitern. Um „Rock Challen-



ge“ auf lange Sicht attraktiv zu machen, wird es eine zusätzliche Diskette mit neuen Fragen geben. Mit dem Titel **„Cosmic Bouncer“** darf sich derjenige schmücken, der



mit seinem lebendigen Tennisball 20 aufregende Level geschafft hat. Ein lebender Ball? Exakt! Aus Versehen wurde ein genetischer Strahl aus dem Kosmos auf den Tennisball gerichtet und erweckte ihn zum Leben. In **„Dragon's Lair“** wird die gute alte Zeit der bösen Drachen, todesmutigen Rittern und holden Edelfrauen wieder aufgewärmt. Dirk der Wagemutige muß darin



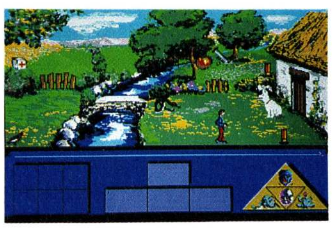
die adelige Daphne aus schmutzigen Drachenklauen befreien. Die Grafik ist an den Stil der Cartoons angelehnt und animiert.

Dicke haben es nicht leicht, denkt **„Chubby Cristle“,** der kugelrunde Parkplatzwächter. Er muß es ja wissen, denn seine mächtige Figur sprengt fast die Uniform. Trotzdem will Chubby nicht abnehmen, im Gegenteil. Ihm schwebt vor, einmal das beeindruckende Gewicht von einer Tonne zu erreichen. So verbringt er die meiste Zeit mit der Suche nach Eßbarem, und zwar möglichst kalorienhaltigem. Falls er dabei einen Fehler macht, drohen ihm schreckliche Strafen; er muß an Gymnastikkursen teilnehmen oder auf einer Gesundheitsfarm schmachten. Nur wer sich mit Plattform- und Leiterspielen auskennt,



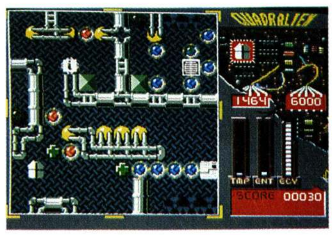
kann den verfressenen Chubby vor diesem Schicksal bewahren.

In **„Indian Mission“** soll Clark Bokel – Professor der okkulten Wissenschaften – auf Götterjagd gehen. Vier blutrünstige Göttergestalten soll der billige „Indiana Jones“-Verschnitt trickreich aus der Welt schaffen. Das Abenteuer beginnt in der Normandie und findet in Indien seinen dramatischen Höhepunkt. Es entbrennt ein haarsträuben-



der Kampf zwischen Professor und Götterschar. Die göttlichen Feinde treten mit übernatürlichen Waffen an. Glücklicherweise hat Professor Bokel auch einiges in petto und greift zu recht unorthodoxen Methoden: Er verwandelt sich im rechten Moment in eine Maus, einen Vogel oder in einen Feuerball. Für 65 Mark (inkl. Porto) bei Müller.

Logotron lädt mit **„Quadralien“** zu einer Doppelmission ein. Astra, der Kernspaltungssatellit vom Jupiter ist überhitzt. Der Spieler muß die Explosion verhindern und zusätzlich die Wurzel des Übels finden und zerstören: die Quadralien-Mutter. Wer sich dieser Aufgabe stellen will, bestellt bei Rushware. Preis: 60 Mark.



„Vyper“ von TopDown ist wieder eines dieser „Shootdown“-Ballerspiele. Man kann sagen, was man will, aber diese Art von Zeitvertreib ist doch immer wieder amüsant und kurzweilig. Bei „Vyper“ muß man sich mit seinem kleinen Raumschiff durch über 100 Level kämpfen. Vorsicht vor den Angriffen der Yarks! Da hift nur geschicktes Ausweichen, denn die Feuerkraft des nur in der horizontalen Ebene manövrierbaren Raumschiffes reicht bei weitem nicht aus, um alle Yarks gleich am oberen Bildschirmrand zu erledigen. Es ist fast unvermeidlich, daß es einen hin und wieder erwischt, aber nach jedem dritten Level, den man überstandenen hat,



gibt's ein kleines Raumschiff als Bonus. Wer alle 100 Level schaffen will, muß einige Tages- und Nachtstunden opfern, um sich mit den verschiedenen Angriffsarten der Yarks vertraut zu machen: Diese Biester kommen nämlich nicht nur von vorn! Bemerkenswert ist die schnelle, flimmerfreie und farbige Animation der Sprites. Ein derart unkompliziertes Spiel zeigt Amigas Möglichkeiten in grafischer Darstellung und schneller, flimmerfreier Animation. „Vyper" hilft dem Spieler, aufgestaute Aggressionen auf eine für die Umwelt friedliche Weise abzuladen. Für zirka 80 Mark bei Ariolasoft.

Mit Logotrons **„Starray"** startet der Spieler in sieben wundersame Welten. Alien-Raumschiffe greifen an und fordern dem Spieler kühnste Flugmanöver ab. Während



des blitzschnellen Scrollings verschieben sich die Ebenen, aus denen eine Welt zusammengesetzt ist. Es entsteht der Eindruck wirklicher Raumtiefe.

In **„Phantasm"** schwirrt der Spieler mit dem Raumgleiter über eine fremdartig bebaute Planetenoberfläche. Es muß ständig auf angreifende Flieger und speziell bezeichnete Bodenziele geschossen werden. Detaillierte Vektorgrafik und Beweglichkeit machen das Spiel für all die, die noch immer am Ballern Spaß finden, interessant genug. Nichts für Anspruchsvolle.

Im Plattformenspiel **„Bionic Commando"** soll die Welt vor den Zargons gerettet werden. Die Grafiken entsprechen allerdings nicht dem Ernst der Situation. Es handelt sich hier um ein eher amüsantes, als dramatisches Spiel.

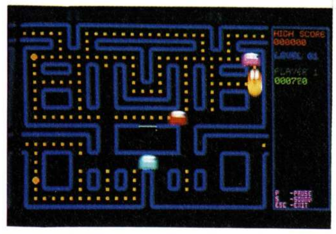


Ace hat den SEGA-Hit **„Alien Syndrome"** auf den Amiga umgesetzt. Ort der Ereignisse ist eine Raumstation. Ein Unfall bei gentechnischen Experimenten hat zu aggressiven, madenförmigen Mutanten geführt, von denen jetzt Wissenschaftler gefangengehalten werden. Ricky und Mary suchen nach den gefesselten Forschern, befreien sie und schießen dabei auf die prallen Riesenmaden. Wenn sie ihre Feuerkraft verstärken wollen, müssen sie aus gekennzeichneten Fächern Ex-



trawaffen nehmen. Feuerbälle, Laser und Flammenwerfer stehen zur Verfügung. Am Ende der Level wartet jeweils ein besonders widerliches Monstrum auf die beiden Helden. Soweit in Ordnung. Unangenehm fällt lediglich das Scrolling auf. Es folgt den Helden zwar in acht Richtungen, doch in sehr ruckartigen Bewegungen.
Capcom läßt wieder im Karatefieber schwitzen. In **„Street Fighter"** tritt der Spieler gegen Kämpfer verschiedener Nationalitäten an und zwar vor landestypischen Straßenszenerien. In England ist der Angreifer ein Punker, in den USA ein Blondkopf vom Typ Dolph Lundgren. Die Japaner bringen einen springfreudigen Glatzkopf, die Chinesen einen wendigen Kämpfer in blauer Kluft. Alle Kämpfer haben offensichtlich dieselbe Karateschule besucht. Ihre Kampfstile unterscheiden sich überhaupt nicht. Jeder führt unkoordinierte Saltos, Sprünge und Schläge aus. Wer schließlich am Boden liegt, hat die Runde verloren.

„Foot Man" von Top Down ist der soundsovielte Aufguß des beliebten Spiels „Pac Man". Doch an Würze verliert es dadurch nicht. Noch immer fasziniert das Kugelwesen, das zielbewußt durchs Labyrinth stiefelt, Punkte sammelt und Gespenstern ausweicht. Wer bei diesem Sport 50 Labyrinthe zu langweilig findet oder damit nicht zurechtkommt, kann sich mit dem Maze Editor selber einen Irrgarten basteln. Einzelteile werden vom Menü gestellt.



In **„Ebonstar"** von Micro-Illusions findet ein intergalaktischer Wettkampf statt, wie er einmal im Jahre 3000 abgehalten werden könnte. Dabei gibt es keine Hindernisse im üblichen Sinne, sondern ein künstliches schwarzes Loch, das ungeschickte Flieger in seine Tiefen saugt.



Wer nicht weiß, was er seinem computerbegeisterten Sprößling oder Freund zu Weihnachten schenken soll, kann sich geschickt aus der Affaire ziehen. Dazu gibt es jetzt Gutscheine bei GTI. Die Gutscheine zu je 50 Mark können dort bezogen und später gegen Software und Zubehör eingelöst werden.

Füchsin im Bikini

Eine Urwald-Lady sucht Hilfe beim Kampf gegen fliegende und krabbelnde Ungeheuer. Reaktionsschnelle Spieler sollen „She Fox" sicher durch Wald und Höhle geleiten.

„She Fox" basiert auf einer ähnlichen Story wie Edgar Rice Borough's Tarzan. Den bekannten Urwaldschreihals haben laut Legende die Affen großgezogen. Auch Heldin „She Fox" wuchs unter der Obhut wilder Tiere auf. Als kleines Mädchen ist sie auf dem von Sauriern beherrschten Planeten Granath ausgesetzt worden. Füchse finden sie in der Wildnis und nehmen sich ihrer an. Da sie auf dem gefährlichen Planeten kaum Überlebenschancen hätte, beschließt der Weisenrat der Füchse, sie mit magischen Kräften auszustatten. So ist sie in der Lage, sich bei Gefahr in einen Fuchs zu verwandeln. Mit einer zauberkräftigen Peitsche kann sie Monstren und Saurier vernichten. Wer nach dieser faden Story ein ebenso langweiliges Computerspiel erwartet, wird überrascht sein. Ein perfekt animierter Sprite, ruckfreies horizontales Scrolling und mitreißende Sounds machen „She Fox" zum gelungenen Hüpf- und Rennspiel. Da die Heldin nach dem Videodigitalverfahren animiert wurde, sind ihre Bewegungen erstaunlich realistisch. Wie Vetter Tarzan springt, rennt und krabbelt sie durch den gefährlichen Dschungel und sucht Bonussymbole. Riesenspinnen, Flugsaurier und andere Ungetüme greifen von allen Seiten an. Ein Peitschenschlag im richtigen Moment löst die Angreifer in Luft auf. Doch werden die Biester von Level zu Level unempfindlicher gegen den magischen Peitschenknall. Dann hilft nur noch die Megapeitsche, von der je ein Exemplar in jedem Bonuslevel verborgen ist. Nach jedem zweiten Urwald-Level vollzieht sich eine Metamorphose. „She Fox" verwandelt sich in einen Fuchs und geht im Höhlenlabyrinth auf Bonusjagd. Die Zeit ist knapp bemessen, und es ist schwierig, alle Symbole einzusammen. Wer sein Ziel — den Fuchsbau — schnell erreicht, erhält für jede verbliebene Sekunde einen Bonuspunkt. Mit Hilfe von Megapeitsche und Edelsteinbonus soll „She Fox" den folgenden Level lebend überstehen. Zur Belohnung gibt es Super-Gems, die eine Multiplikation der Punktzahl bedeuten. Doch wenn die Fuchslady in's Wasser stürzt oder einem Saurier zum Opfer fällt, sind sowohl Megapeitsche als auch eins von zehn Fuchsleben dahin. Ist auch aus der Anleitung kein Grund für die wilde Hatz ersichtlich, so bringt „She Fox" doch eine Menge Spaß und bereichert jede Spielesammlung.

(mms)

Name: She Fox	
Preis: 70 Mark	
Hersteller: Martech	
Vertrieb: Rushware	
Spaß	10
Grafik	9
Sound	8
Idee	4
Schwierigkeit	6



Die Fuchslady holt zum Schlag gegen den Urwaldschrat aus. Jeder Fehlschlag kostet ein Leben.



„She Fox" hat sich in einen Fuchs verwandelt. Nur in dieser Gestalt kann sie die Megapeitsche finden.



Fliegende Monster verfolgen das Mädchen über Abgründe, Flüsse und Palmenhaine. Ein Peitschenschlag tötet den Angreifer.

Impressum

Chefredakteur: Manfred S. Schmidt (mss) (verantwortlich, Anschrift siehe Redaktion)
Redaktion: Uwe Knierim (uk), Mike Nickles (mn), Ute Bahn (ub)
Ständige Mitarbeiter: Ilse und Rudolf Wolf (iw, rw), Mathias M. Schmidt (mms)
Redaktionsassistentin: Siggi Pöschel (sp)
Gestaltung: Karin Wirth

Anschrift der Redaktion: CW Publikationen (Amiga-Welt), Postfach 40 04 29, Rheinstraße 28, 8000 München 40, Telefon: 0 89/3 60 86-0, Telex: 5 215 350 comw d, Telekopierer: 0 89/3 60 86-1 09

Auslandsredaktionen:
Österreich: COMPUTERWELT Österreich, CW-Publikationen Verlagsgesellschaft mbH, Zieglergasse 6/1, A-1070 Wien, Tel.: (00 43) 222/93 05 08
Schweiz: COMPUTERWORLD Schweiz, Witikonerstr. 15, Postfach 253, CH-8030 Zürich, Tel.: (01)55 10 77, Telex: 816 710 cw cich
USA: CW-COMMUNICATIONS INC., 375 Cochituate Road, Box 9171, USA-Framingham, Mass. 01701, Tel.: 001/617/879 07 00, Tx.: 00230/951 153 computwrld fmh
Japan: COMPUTERWORLD/Japan, 1- 19-7, Shintomi Chuoku, J-Tokyo 104, Tel.: 00 81/3/5 51 38 82
China: China COMPUTERWORLD, 74 LuGuGun Road, P.O. Box 750, RC-Beijing 100039, Tel.: 00 88/6/814-61 74
Verlagsrepräsentanten:
Großbritannien: Beere Hobson Ass., Euan C. Rose, 34 Warwick Road, Kenilworth, GB-Warwickshire CV8 1HE, Tel.: 00 44/926/51 24 24.
USA: CW International Marketing Services, Diana La Muraglia, 375 Cochituate Road, Box 880, USA-Framingham, Mass. 01701, Tel.: 001/617/879 07 00, Tx.: 00230/951 153 computwrld fmh
Taiwan: Alice Chen ACE Media, Agency Co., Ltd., P.O.Box 26-578, Taipei R.O.C., Tel. 021721-4302, Telex 14142 acegroup
Japan: Steven Yamada, Tokyo Representative Corp., Sanshin Kogyo Jimbocho 3F, Chiyoda-ku, Tokyo 101 Japan, Tel.: 230-4117, Tlx.: (781) J26860
Einsendungen: Manuskripte werden gerne von der Redaktion entgegengenommen. Die Zustimmung zum Abdruck und zur Vervielfältigung wird vorausgesetzt. Gleichzeitig versichert der Verfasser, daß die Einsendungen frei von Rechten Dritter sind und nicht bereits an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten wurden. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Sonderheft-Bestellungen: Direkt beim Verlag, Abt. Vertrieb (Anschrift s. u.) gegen Verrechnungsscheck + DM 2,– Porto oder per Nachnahme sowie über den Buchhandel
ISSN 0933-8616

Vertrieb Handelsauflage: MZV Moderner Zeitschriften Vertrieb GmbH, Breslauer Straße 5, 8057 Eching, Tel.: 089/31 90 06-0, Telex: 522 656.
Telefon-Durchwahl: 089/3 60 86 ( )
Vertriebsleitung: Brigitte Schleibinger (-155/-156)
Anzeigenleitung: Sylvia Stier (-161); (verantwortlich für Anzeigen, Anschrift siehe unter Anzeigen)
Anzeigenverkaufsleitung:
Barbara Schönberger (-130)
Anzeigendisposition: Ursel Sauter (-126)
Anschrift für Anzeigen und Vertrieb: Postfach 400 429, Rheinstraße 28, 8000 München 40, Telefon: 0 89/3 60 86-0, Telex: 5 215 350 comw d, Telekopierer 089/3 60 86-1 09
Disposition:
Rainer Oberländer (Leitung), Stefan Liba
Druck und Beilagen:
Carl Gerber Grafische Betriebe, Muthmannstraße 4, 8000 München 45, Tel.: 089/3 23 93-233 (Anschrift für Beilagen)
Zahlungsmöglichkeiten:
Bayerische Vereinsbank, BLZ 700 202 70, Konto-Nr. 116 000, Pschk. München 97 40-800
Erfüllungsort, Gerichtsstand: München
Verlag: CW-Publikationen Verlagsgesellschaft mbH, Rheinstraße 28, 8000 München 40, Telefon: 089/3 60 86- 0, Telex: 5 215 350 comw d, Telekopierer: 089/3 60 86-109
Art Director: Darinka Bratusa
Produktionsleitung: Heinz Zimmermann
Redaktionsdirektor: Dieter Eckbauer
Kaufm. Leitung: Jürgen T. Reischl
Geschäftsführer: Eckhard Utpadel, Walter Boyd, Patrick McGovern

CW PUBLIKATIONEN

Ihre starke Verbindung zur Computerwelt

Amiga-Welt ist eine Publikation der IDG Communications, der Welt größter Herausgeber für computerbezogene Informationen. IDG Communications veröffentlicht über 90 Computer-Publikationen in 33 Ländern. Jeden Monat lesen vierzehn Millionen Menschen eine oder mehrere Publikationen von IDG Communications. Alle Publikationen sind dem IDG International News Service angeschlossen, der die neuesten Meldungen aus der Kommunikations- und Informationstechnologie für die Redaktionen bereithält. IDG Communications-Publikationen sind:
ARGENTINIEN: Computerworld Argentina; ASIEN: Communications World, Computerworld Hong Kong, Computerworld Malaysia, Computerworld Singapur, PC Review, Computerworld SE Asia; AUSTRALIEN: Computerworld Australia, Communications World, Australian PC World, Australian Macworld; BRASILIEN: Data News, PC Mundo, Micro Mundo; BUNDESREPUBLIK DEUTSCHLAND: Computerwoche, Information Management, PC Woche, PC Welt, MS-DOS-Welt, Amiga Welt; CHILE: Informatica, Computacion Personal; DÄNEMARK: Computerworld Danmark, PC World Danmark; ENGLAND: Computer News, DEC Today, ICL Today, Lotus, PC Business World; FINNLAND: Tietoviikko, Mikro; FRANKREICH: Le Monde Informatique, Distributique, Info PC, Telecoms International; GRIECHENLAND: Micro and Computer Age; INDIEN: Dataquest; ISRAEL: People & Computers Weekly, People & Computers Biweekly; ITALIEN: Computerworld Italia; JAPAN: Computerworld Japan; KANADA: Computer Data; MEXICO: Computerworld Mexico; NIEDERLANDE: Computerworld Nederland, PC World Benelux; NEUSEELAND: Computerworld New Zealand; NORWEGEN: Computerworld Norge, PC World Norge; ÖSTERREICH: Computerwelt Österreich; SAUDI ARABIEN: Arabian Computer News; SÜD-KOREA: Computerworld Korea, PC World Korea; SPANIEN: Cimworld, Computerworld Espana, Commodore World, PC World Espana, Communicaciones World, Informatica Industrial; SCHWEDEN: Computer Sweden, MikroDatorn, Svenska PC World; SCHWEIZ: Computerworld Schweiz; UNGARN: Computerworld SZT, PC Mikrovilag; VENEZUELA: Computerworld Venezuela; VEREINIGTE STAATEN VON AMERIKA: Amiga World, CD-ROM Review, CIO, Computer Currents, Computerworld, Computers in Science, Digital News, Federal Computer Week, 80 Micro, Focus Publications, InCider, Infoworld, Macintosh Today, Macworld, Computer + Software News (Micro Marketworld/Lebhar-Friedman), Network World, PC World, Portable Computer Review, Publish!, PC Resource, Run, Windows; VOLKSREPUBLIK CHINA: China Computerworld, China Computerworld Monthly

IDG
An IDG Communications Publication

Inserentenverzeichnis

Inserent	Seite
A. + L. Meier-Vogt	**101**
ALCOMP	**81**
Amigaoberland	**7**
Atlantis Vertriebsges. mbH	**27**
High Speed Software Blanke	**71**
CSV Riegert	**71**
CWTG — Computerversand Tiede	**63**
Combitec GmbH i. Gr.	**12, 13**
D. Kabs & F. Winterscheid	**63**
Computergrafik-Verlag Lechner	**17**
CompuStore Handelsges. mbH	**23, 113, 115**
DISC COMPANY EUROPE	**11**
Data Becker GmbH	**25, 39, 137**
Flesch & Hörnemann GbR	**43**
GFA Systemtechnik GmbH	**2.US.**
GIGATRON	**55**
CIT — Wiese	**19**
GTI Grennville Trading & Import GmbH	**105**
Siggis Software Shop	**63**
Gnoth, D.	**105**
Hieske, D.	**73**
IDEE-Soft I. Dinkler	**55**
Intelligent Memory	**3. US.**
IrseeSoft SPCS	**23**
Kingsoft	**4. US.**
Korona Soft	**105**
Merkens EDV	**53**
MessTech GmbH	**55**
NEC Deutschland GmbH	**9**
Ossowski, S.	**93**
PBC Biet	**41**
PMD M. Aures	**23**
Peekhaus, F.	**53**
Philgerma GmbH & Co. KG	**10**
Rat und Tat GmbH	**93**
Schmielewski, U.	**23**
Ruhrsoft Scheer	**93**
Software 2000	**71**
Soft Mail AG	**71**
Softwareversand Müller	**105**
Stalter Computerbedarf	**63**
TFM & Partner	**105**
technicSupport	**57**
INFOMARKT	**S. 117–119**

Amiga-Shop Ruth, Amiga-Sofware Küppers, C. V. S. Home Computer Laden, CIK-Computertechnik Klepsch, COMPEDO, CPS-Computertechnik GmbH, Dombrowski, R., Fischer, A., Funkcenter Mitte GmbH, HK-Computer, Harders, K., Heidrich, D., Hofstede, T., Keim, P., Kirschbaum Medienberatung, Microtec, Musik- Grafiksoftware Shop Czerny, OPTIVISION, Prolinea Computerprogramme, SWS Computersysteme GdbR, Schellhammer, Schramm, K.-M., Technocom, Witte's Digitaltechnik, computer corner Duffer

Gewerbliche Gelegenheiten	**S. 120**

Flirt — und mehr — im Glaspalast

„Erleben Sie die romantischsten Treffen, die Sie sich vorstellen können und Begegnungen, von denen Sie kaum gewagt haben zu träumen!" So verheißungsvoll klingt die Einladung zum Erwachsenenspiel „Romantic Encounters at the Dome". Wer diese marktschreierische Anpreisung schätzt, findet sicher Gefallen am Gebotenen. Also schnell die beiden Spieldisketten laden, den persönlichen Code eingeben und einen Partner aussuchen. Bei der Partnerwahl eröffnen sich unerwartete Perspektiven. Man hat die freie Wahl zwischen Männern, Frauen und sogenannten „Anderen". Diese ominösen „Anderen" regen die Phantasie enorm an. Erwartungsvoll stürzt man sich in's Getümmel und pirscht in fünf Stockwerken nach Liebeswilligen. Da darf nach allen Regeln der Verführungskunst geflirtet und „angemacht" werden. Dann geht es zur Sache. Im Penthouse des Vergnügungsdomes erwartet den erfolgreichen Aufreißer der sprichwörtliche Himmel. Wer nun ganz genau wissen will, wie es um seine erotischen Qualitäten bestellt ist, fährt nach bestandenen Abenteuern schnurstracks in die psychologische Abteilung. Dort bekommt er einen Stapel Fragebögen und muß lediglich zutreffende Antworten ankreuzen. Fragen als auch Antworten sind derart, daß sie Möchtegern-Playboys, wie etwa pubertierenden Computerfreaks und midlife-criselnden Rechtsanwälten, die Röte in's Gesicht treiben können. Die Textvorgaben für adventurelustige Damen haben dieselbe Wirkung. Dennoch ist dies nicht Ziel der Befragung. Vielmehr wird aus den angekreuzten Antworten ein psychologisches Gutachten errechnet. Darin kann jeder nachlesen, ob er zu den forschen, coolen Typen, den zartbesaiteten, romantischen Träumern oder zum ordinären Durchschnitt gehört. Von Abenteuer und Gutachten sollte man allerdings nicht zuviel erwarten, denn 90 Prozent aller Annäherungsversuche enden in Frustration oder Trunkenheit, — so behaupten zumindest die beiden Psychologen, die an der Entwicklung des Spiels beteiligt waren. Entsprechend dieser Annahme steht hier nach vollbrachten Taten ein wüstes Trinkgelage an. Die Erfolgreichen begießen ihre Eroberung, die Frustrierten spülen ihren Seelenschmerz herunter. Vielmehr bleibt auch dem Spieler dieses pseudoerotischen Adventures nicht übrig, denn er hört weder Musik noch sieht er Grafik. Schade, zumal der Amiga sich eher zur Grafik- und Musikdarstellung eignet als zur digitalen Kompensation eines unausgefüllten Liebeslebens.

(R.Y. de Flers)

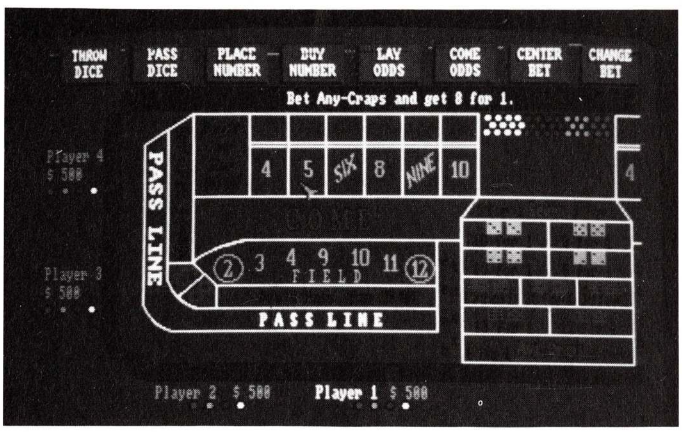
Name: Romantic Encounters at the Dome
Preis: ca. 80 Mark
Hersteller: MicroIllusions
Vertrieb: Ariolasoft

Spaß keiner
Grafik keine
Sound keiner
Idee 1
Schwierigkeit 3

Würfelfieber

Würfel klackern gegen die Bande. Gebannt erwarten die Spieler den Wert ihres Wurfes. Ein Hauch von Atlantic City, Las Vegas und Reno umgibt die vier Glücksspieler am Amiga. Das grüne Spielfeld ist mit einem Felder- und Zahlenwirrwarr bedeckt und erinnert stark an Roulette. Bevor man nun Haus und Hof verspielt, sollte man sich im 19seitigen Handbuch mit Spielregeln und Bedienung vertraut machen. Spielaktionen werden samt und sonders mit der Maus im Menü angeklickt. Bis auf die Installation des Programms und die Eingabe der Mitspieler wird die Tastatur nicht benutzt. Beim Installieren wird eine beträchtliche Anzahl Parameter eingestellt. Die Farbenbestimmung erinnert stark an DeluxePaints RANGE-Auswahl und erlaubt fantasievolles Colorieren des Craps-Feldes. Spieler von der ungeduldigen Sorte wählen im Menü den Very-Fast-Modus. So müssen sie nicht zitternd warten, bis das ganze Spielfeld nach links abgescrollt ist, um die Würfel gegen die Bande schlagen und dann vom Dealer auf das Dealerfeld zurückschieben zu lassen. Wer sich während des Spiels um Regeln, Gewinnchancen und Spielstatistik informieren will, findet Rat im Hilfsmenü. In der Abteilung Statistik wird wirklich alles erfaßt, selbst die Würfelumdrehungen. Craps-Regeln gibt es in den Varianten Reno, Las Vegas und Atlantik City. Neben dieser spielerischen Vielfalt ist das Programm aber auch eine „Academy", die verschiedene Strategien lehrt. Glücksspieler sind also gewappnet, sollte es sie irgendwann einmal an einen amerikanischen Spieltisch verschlagen. Unser Testexemplar hatte lediglich zwei Schwächen. Chips ließen sich mit dem Mauszeiger sehr schlecht fassen, und wenn man die Chipstapel zu oft anklickte, stürzte das Spiel ab. Nun bleibt nur ein Rat: Aufhören, bevor der Amiga verspielt ist.

(R. de Flers)



An einem solch harmlos aussehenden Spieltisch hat schon mancher Spieler sein letztes Hemd verloren.

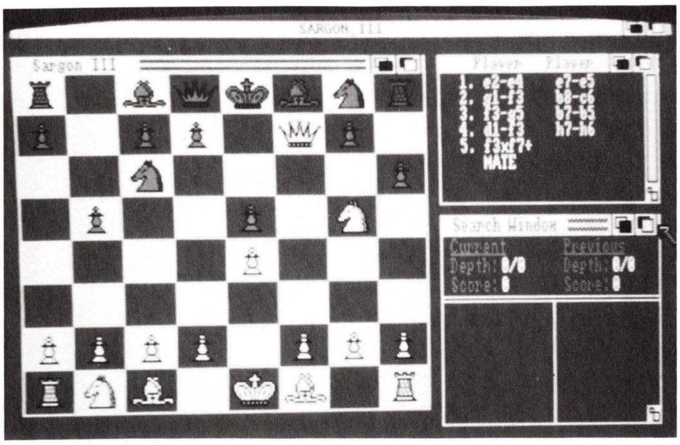
Name: Craps Academy
Preis: ca. 80 Mark
Hersteller: Microillusions
Vertrieb: Ariolasoft

Spaß 7
Grafik 6
Sound 3
Idee 5
Schwierigkeit 6

Exquisiter Denksport

„Sargon III", der Meister des königlichen Spiels, sucht Gegner und Schüler. Logotrons neues Schachprogramm kennt 68 000 Züge und beherrscht neun Schwierigkeitsstufen. Damit hat es als Spielpartner und als Schachlehrer eine Menge zu bieten. Zusätzlich überwacht und notiert es fleißig den Spielstand, weiß in scheinbar ausweglosen Spielsituationen Rat und gestattet dem Schachlehrling, einen unbesonnen Zug zurückzunehmen. Damit eignet sich das Spiel für Anfänger, Fortgeschrittene und Meister gleichermaßen. Entsprechend bietet das Handbuch jeder Spielerkategorie Nützliches. Anfänger lernen die Figuren und deren Züge im ersten Kapitel genau kennen. Im zweiten Kapitel wird erklärt, wie man mit „Sargon III" umgeht. Anschließend werden den Spezialisten über 100 historische Partien vorgestellt, die alleamt im Great-Games-Speicher konserviert sind. Die Abteilung „Problemschach" behandelt kniffelige Konstellationen wie Eröffnung, Endspiel und Schachmatt. Auf die Analyse des Problems folgen Tips und mehrere Lösungsstrategien. „Sargon III" hilft als Ratgeber bei der praktischen Problembewältigung oder stellt sich als Gegner zur Verfügung. Wer darauf dankend verzichtet und lieber mit einem Menschen wetteifert, kann ein besonderes Feature in Anspruch nehmen: „Veriy" aus dem Menü Options macht das Programm zum Schiedsrichter einer Partie. Wenngleich „Sargon III" nicht eigentlich als Schiedsrichter gedacht ist, so sorgt doch das Spiel eines Menschen gegen einen anderen für das Erfolgserlebnis, das die Maschine ihrem Gegner meist verwehrt.

(mms)



Klassische Schachkonstellation: Der „Schäferzug".

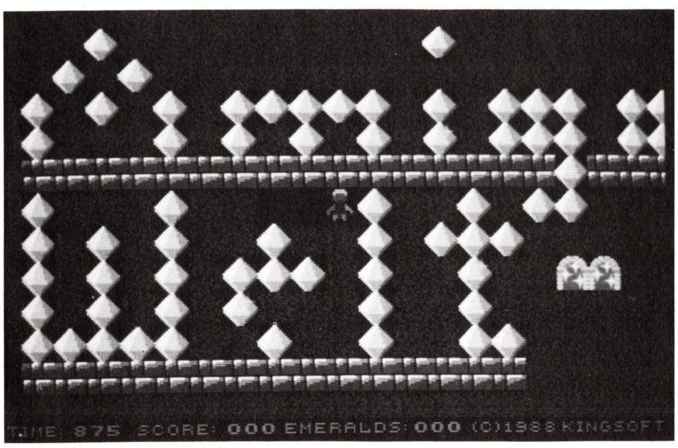
Name: Sargon III
Preis: ca. 80 Mark
Hersteller: Logotron
Vertrieb: Rushware

Kategorie	Wertung
Spaß	8
Grafik	3
Sound	5
Idee	7
Schwierigkeit	10

Edles Spiel mit edlen Steinen

Hektisch sammelt der kleine Schatzsucher funkelnde Smaragde auf. Auf der Suche nach neuen Fundorten flitzt er durch das unterirdische Labyrinth. Es ist ein Wettrennen mit der Zeit, denn jeder Level muß innerhalb eines festgesetzten Zeitlimits durchquert und leergesammelt werden. Mauern, Monster und verschlossene Türen erschweren das Vorwärtskommen ungemein. Aber als echter Edelsteinsucher hat unser Männlein Sprengstoff dabei und reißt steinerne Hindernisse nieder. Wenn die Höhlenmonster Schwierigkeiten machen, werden sie kurzerhand erschlagen. Massive Türen lassen sich nur mit einem geeigneten Schlüssel öffnen. Dies ist das Prinzip eines der meistverkauftesten Spiele: „Emerald Mine". Kingsofts Programmierer fühlten sich durch den Erfolg herausgefordert, ein noch attraktiveres Spiel zu entwickeln. Dieser Anforderung soll „Emerald Mine II" entsprechen. Das Spielprinzip ist mit dem des ruhmreichen Vorgängers identisch, dessen kleine Fehler in der neuen Version ausgebügelt wurden. Hinzugekommen sind 80 neue Level und ein komfortabler Editor, mit dem sich unzählige neue Level schaffen und vorgegebene Level variieren lassen. So sind Phantasie und Kreativität keine Grenzen mehr gesetzt. Um schöpferische Spieler anzuspornen, hat Kingsoft einen Wettbewerb ausgeschrieben. „Wer bastelt das beste Emerald-Spielfeld?" lautet das Motto. Die gelungensten Levels werden anschließend unter dem Titel „Emerald Mine III" zu einer Siegerdiskette zusammengestellt. Diese dritte Version wird hoffentlich kürzere Ladezeiten haben, denn die bisherige Ladezeit von eineinhalb Minuten liegt hart an der Toleranzgrenze.

(mms)



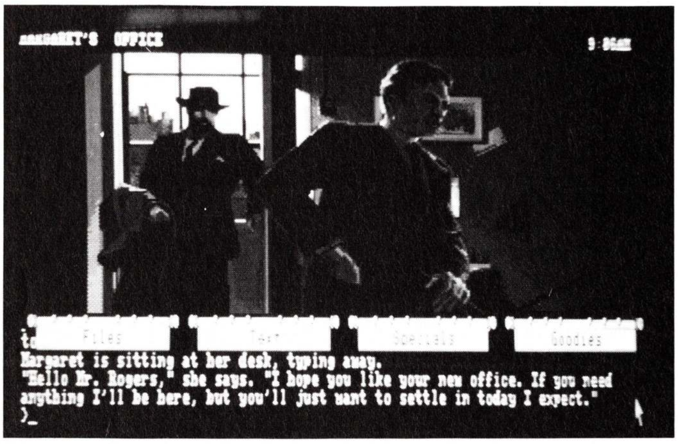
Selbstgebastelte Spielfelder können mit phantasievollen Mustern versehen werden.

Name: Emerald Mine II
Preis: 50 Mark
Hersteller/Vertrieb: Kingsoft

Kategorie	Wertung
Spaß	10
Grafik	6
Sound	8
Idee	7
Schwierigkeit	10

Geld ist Macht

Rainbirds neues Adventure „Corruption“ macht den Spieler zum Yuppie. Sein Name: Derek Rogers. Der junge Mann ist erfolgreich und glaubt das große Los gezogen zu haben, als er Teilhaber einer Großfirma wird. Zunächst verspricht dieser Einstieg tatsächlich Reichtum und Macht, doch findet Derek bald heraus, daß im Big Business nicht nur mit legalen Mitteln gearbeitet wird. Dereks Partner David ist nämlich neben seiner ehrenhaften Tätigkeit für die Firma in dunkle Geschäfte verwickelt. Derek ist dabei im Wege. Er muß um sein Leben fürchten. Zu allem Übel hält auch seine Frau nicht mehr zu ihm. Sie trägt sich mit Scheidungsabsichten.



Aufsteiger Rogers macht sich mit seinem neuen Arbeitsplatz vertraut. Hier laufen die Fäden des Big Business zusammen.

Der „Corruption“-Spieler soll all diese Probleme lösen. Viel Zeit bleibt jedoch nicht, — die eingeblendete Uhr tickt gnadenlos. Verabredungen im Spielcasino, Termine und Börsengeschäfte verlangen Pünktlichkeit. Bei diesen gesellschaftlichen und beruflichen Treffen gibt es Interessantes zu erfahren, während man sich zwanglos mit den Anwesenden unterhält und aufmerksam deren Reaktionen auf bestimmte Themen beobachtet. Allerdings sind die Gespräche ein wenig schwerfällig, da der Spieler seine Fragen und Antworten nach einem festgelegten Eingabemuster formulieren muß. Ein Parser mit umfangreichem Wortschatz gleicht diesen Mangel aus. Zudem ist die Bedienung des Programmes sehr komfortabel. Häufig benötigte Kommandos können auf die Funktionstasten gelegt werden. Die Größe der Buchstaben ist variabel, Texte können ausgedruckt oder als Sprache ausgegeben werden. Wer der Lösung nicht näher kommt, erhält Hilfe, wenn er einen bestimmten Code eingibt. Weitere Hinweise geben Dereks beiliegendes Notizbuch und die Bandaufnahme eines verhängnisvollen Gespräches.

(A. Peters/U. Loos)

Name: Corruption
Preis:n.n. bekannt
Hersteller: Rainbird
Vertrieb: Rushware

Spaß	9
Grafik	9
Sound	8
Idee	9
Schwierigkeit	9

Bezugsquellen

(ohne Anspruch auf Vollständigkeit)

Aegis Developement
Deutscher Vertrieb: Atlantis, Dunantstr. 53, 5030 Hürth, Tel.: 0 22 33/ 41 08 1;
CompuStore, Fritz Reuter Str. 6, 6000 Frankfurt 1, Tel.: 0 69/ 56 73 99
DTM, Poststr. 25, 6200 Wiesbaden, Tel.: 0 61 21/ 56 00-84;
GTI GmbH, Zimmersmühlenweg 73, 6370 Oberursel, Tel.: 0 61 71/ 36 25 22,
Jumbo Soft, Horemansstr. 2, 8000 München 19, Tel.: 0 89/ 1 23 40 65
Leisuresoft, Industriestr. 23, 4709 Bergkamen 5, Tel.: 0 23 89/ 60 71,
Soyka, Hattingerstr. 685, 4630 Bochum 5, Tel.: 0 23 4/ 4 98 25-26

Associated Computer Service
1306 East Sunshine, Springfield, MO 65805, Tel.: U.S.A. 417/ 887-7373

A-Squared Distributions
6114 La Salle Ave, Suite 326 Oakland, CA 94611, Tel.: U.S.A 415/ 339-0339

Brown-Wagh Publishing
Deutscher Vertrieb: Atlantis (s. o.), CompuStore (s. o.), DTM (s. o.), GTI (s. o.), Jumbosoft (s. o.), Soyka (s. o.)

Byte by Byte
Deutscher Vertrieb: Atlantis (s. o.), CompuStore (s. o.), DTM (s. o.), GTI (s. o.), Jumbosoft (s. o.), Soyka (s. o.)

Eagle Tree Software
Deutscher Vertrieb: Atlantis (s. o.), Leisuresoft (s. o.) GTI (s. o.), Jumbosoft (s. o.)

Electronic Arts
Deutscher Vertrieb: Atlantis (s. o.), CompuStore (s. o.), DTM (s. o.), GTI (s. o.), Intelligent Memory, Borsigallee 18, 6000 Frankfurt 60, Tel.: 0 69/ 41 00 71, Soyka (s. o.)

Hash Enterprises
Deutscher Vertrieb: Atlantis (s. o.), DTM (s. o.), GTI (s. o.)

Impulse Inc.
Deutscher Vertrieb: Atlantis (s. o.), CompuStore (s. o.), DTM (s. o.), GTI (s. o.), Jumbosoft (s. o.),

Lamm Computersysteme
Schönbornring 14, 6078 Neu-Isenburg 2, Tel.: 0 61 02/ 5 25 35

Magni
9500 SW Gemini Drive, Beaverton, OR 97005, Tel.: U.S.A. 5 03/ 6 25-84 00 oder 8 00/ 23 7-59 64

Merkens EDV
Fuchstanzstr. 6 A, 6231 Schwalbach, Tel.: 0 61 96/ 3026

Micro Magic
Deutscher Vertrieb: Atlantis (s. o.), CompuStore (s. o.), DTM (s. o.), GTI (s. o.)

Mindware International
Deutscher Vertrieb: Atlantis (s. o.), DTM (s. o.), GTI (s. o.), Jumbo Soft (s. o.) Soyka (s. o.)

Mimetics
Deutscher Vertrieb: Atlantis (s. o.), GTI (s. o.), Intelligent Memory (s. o.)

NewTek, Inc.
Deutscher Vertrieb: Atlantis (s. o.), CompuStore (s. o.), DTM (s. o.), GTI (s. o.), Jumbosoft (s. o.), Leisuresoft (s. o.) Soyka (s. o.)

PBC Peter Biet
Georg-Fischer-Str. 5, 6415 Petersberg 2, Tel.: 0661/801263

Progressive Peripherals & Software
Deutscher Vertrieb: Atlantis (s. o.), Compustore (s. o.), DTM (s. o.), GTI (s. o.), Jumbosoft (s. o.), Soyka (s. o.)

PVS Publishing
Deutscher Vertrieb: Atlantis (s. o.), Lamm (s. o.), Videocomp (s. u.)

Right Answers Group
Deutscher Vertrieb: Atlantis (s. o.), CompuStore (s. o.), GTI (s. o.), Jumbosoft (s. o.)

SunRize Industries
Deutscher Vertrieb: CompuStore (s. o.), DTM (s. o.), Intelligent Memory (s. o.)

Syndesis
Deutscher Vertrieb: Atlantis (s. o.), GTI (s. o.), Jumbosoft (s. o.)

Videocomp
Berger Straße 193, 6000 Frankfurt 60, Tel.: 0 69/ 46 71 01



Die Datenbank mit Profil.

DATAMAT hat ganz gehörig an Profil gewonnen: Die neue professionelle Version des beliebten Amiga-Programms läßt sich problemlos bedienen wie eine Dateiverwaltung, bietet aber die enormen Vorteile einer relationalen Datenbank.

Ein Programm mit Profil eben: Profil heißt die über 200 Befehle und Funktionen starke Interpretersprache, die das Programmieren so komfortabel macht.

Anfänger finden mit bequemen Pulldown-Menüs und bis zu acht offenen Fenstern den schnellen Einstieg in die Dateiverwaltung, fortgeschrittene Anwender nutzen die an BASIC angelehnte, strukturierte Programmiersprache. Die ist nicht nur in der leichten Erlernbarkeit ganz groß: Mit „Profil" entstehen eigene Benutzeroberflächen so professionell wie komplexe Auswertungen. Verbindungen zwischen Dateien sind auch ohne Programmierung möglich und - besonders praktisch - eine Änderung dieser Relationen erfordert keine Reorganisation der Dateien.

Das neue Programm beweist zudem im wörtlichen Sinn Größe: Die Dateien können bis zu zwei Milliarden Zeichen umfassen. Bei zwei Milliarden stößt auch erst die Datensatz-Menge an ihre Grenze; die maximale Datensatzgröße beträgt 64.000 Zeichen. Vollkommen unbeschränkt ist die Zahl der Datenfelder (max. Feldgröße 32.000 Zeichen). Und wer aus diesen Unmengen von Daten rasch eine bestimmte Angabe benötigt, der hat sie - bei bis zu 80 Indexfeldern - in Sekundenschnelle zur Hand. Verlagseigene Tests beweisen: Gesuchte Werte oder Texte können in höchstens zwei Sekunden gefunden werden, ausgewählt aus sage und schreibe 10.000 Datensätzen.

Die Vorteile von DATAMAT Professional lassen sich also leicht ausrechnen. Schwierigere Operationen dürfen Sie getrost dem Programm überlassen: Über 100 Funktionen und Konstanten stehen zur Verfügung, um Felder auch mit elaborierten Formeln zu verknüpfen. Was wieder ein schönes Stück zum neuen DATAMAT-Profil beiträgt.
Fordern Sie ganz unverbindlich unser kostenloses Info an.

DATAMAT Professional AMIGA DM 498,-

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

COUPON

☐ Bitte senden Sie mir ganz unverbindlich Ihr kostenloses Info zu DATAMAT Professional AMIGA
☐ Hiermit bestelle ich für DM 498,- DATAMAT Professional AMIGA
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

NAME, VORNAME

STRASSE

ORT

Larry auf Minnepfaden

Alle Frauen wollen von Larry nur das Eine, — daß er sie in Ruhe läßt. Also braucht er einen wirklichen Kenner der zwischenmenschlichen Kommunikation. Kein anderer kann Larry erfolgreich an die Frau bringen.



In zwielichtigen Bars versucht Larry sein Glück.

Sierras Adventure-Reihe „Leisure Suit Larry" gelang es, amerikanische Computerfreunde jeden Alters vor den Bildschirm zu bannen. Jetzt tritt das Erwachsenenspiel „Leisure Suit Larry in the Land of the Lounge Lizards" dieser Serie auch bei uns seinen Siegeszug durch Büros, Clubs und Wohnungen an. Und es sind sicher nicht nur Erwachsene, die sich verbissen durch die fünf Fragen kämpfen, die den Reifegrad jedes Larry-Fans prüfen sollen. Wer nämlich zwei Fragen falsch beantwortet, wird nicht zum Spiel zugelassen. „You're a kid!! Shame on you for trying to deceive a friendly game like this!" empört sich das Programm und rät, ein paar Jahre abzuwarten. Hoffentlich ist in diesem Moment jemand da, der mit Glück und Verstand die richtigen Antworten findet. Manche Option schließt sich allerdings von selbst aus; jeder kann sich denken, daß „IBM" nicht die Abkürzung für „International Branch of the Mafia" ist. Wer alle Antworten verfehlt, verpaßt eine amüsante Geschichte, in der er die Rolle des erfahrenen Salonlöwen spielte und als solcher den trotteligen Larry in allen Lebenslagen beriete. Denn ohne Hilfe ist der Anti-Held verloren. Kein Wunder, wenn man Larry's Vergangenheit betrachtet!

„Ich will hier raus" beschließt Larry eines Tages

Bis vor kurzem hat Larry Laffer das solideste und langweiligste Leben geführt, das man sich vorstellen kann. Er trat seinen Job als Computertechniker pünktlich an, machte pünktlich Feierabend und kehrte heim zur Mutter. Bei schmalziger Musik vertiefte er sich allabendlich in sein Lieblingsbuch „Know Your RS-232 Serial Port" und ging früh zu Bett, . . . bis ihm eines Tages klar wurde, daß er fast 40 Jahre lang am Leben vorbeigelebt hatte. Mit all dem soll jetzt Schluß sein! Larry will Erfahrungen sammeln, Erfahrungen mit Frauen, Glücksspiel und Alkohol.

Kurzentschlossen geht er zum Friseur und läßt sich einen zeitgemäßen Haarschnitt verpassen. Seine biedere Kleidung tauscht er gegen einen nagelneuen, hellen Freizeitanzug und gibt sich mit ein paar Metern Talmischmuck einen halbweltlichen Anstrich. Um das neue Image abzurunden, bekämpft er seinen penetranten Mundgeruch mit einem desodorierenden Spray. Im neuen Outfit und mit ein paar Dollars in der Tasche fährt er in die Metropole der Verwerflichkeit: Lost Wages. Dort will er all die Lasterhöhlen besuchen, vor denen seine Mutter immer gewarnt hat. Hier beginnt für den Berater vor dem Bildschirm ein hartes Stück Arbeit.

Ohne Passwort läuft nichts bei den Frauen

Erste Szene. — Verloren steht Larry vor einer Bar zweifelhaften Rufes. Das ist genau das, was er sucht! Er geht hinein. Aha, hier soll es also tolle Frauen geben. Leider ist nur eine einzige Dame anwesend, und die unterhält sich angeregt mit einem Kerl. Larry unternimmt trotzdem einen schüchternen Annäherungsversuch. Abgeblitzt! Playboy-Lehrling Larry ist scheinbar nicht



Verkehrsrowdies behindern den Weg zum Glücksspiel. Sind denn alle gegen Larry?



Gilt es lange Wege zurückzulegen, darf Larry auch mal ein Taxi benutzen.

in Form. Er bestellt ein Bier und spült seinen Frust damit herunter. Als er aufgestanden ist, entdeckt er eine verschlossene rote Tür. Auf sein Klopfen öffnet sich ein Guckloch. „Yeah. Whatsda paaswoid?" grölt jemand im schlimmsten Dialekt. Larry kennt das Password nicht.

Nun liegt es bei dem Scharfsinn des Spielers, herauszufinden, wie das Password heißt, welche aufregenden Dinge hinter der roten Tür warten und wie es weitergeht. Larry wird unerwartete Abenteuer erleben, wenn sein Berater die verrückten Ideen der Larry-Programmierer nachvollziehen kann und immer das Unglaubliche erwägt. Zudem müssen diverse Utensilien mitgenommen oder erworben werden. Ohne diese Dinge sind bestimmte Situationen nicht zu bewältigen. Man lasse seinen Schützling Larry also alles annehmen und einstecken, was er irgend bekommen kann. Rose, Messer, Prophylacticum und Clubkarte sind nur Beispiele aus der Reihe (lebens-)wichtiger Gegenstände. Wo man diese Utensilien findet, ist der Intelligenz des Spielers überlassen. Schließlich gibt es genügend verschiedene Örtlichkeiten, an denen man fündig werden kann: Bar, Nebenraum, Toilette, Shop, Disco, Casino und Hotel.

Selbst im Straßenverkehr benimmt sich Larry ungeschickt

Kurze Strecken geht man zu Fuß, wobei Vorsicht vor düsteren Gestalten angeraten ist. Größere Entfernungen legt man mit dem Taxi zurück. Allerdings muß man Larry sogar auf dem Weg zum Taxi gängeln, da der unerfahrene Bursche auf die Straße läuft, ohne auf den Verkehr zu achten. Das Spiel fände so ein tragisches Ende. Dennoch müßte man an diesem Punkt nicht verzweifeln. Laut Hilfsmenü — Taste F1 — beginnt das Spiel mit F9 von neuem, ohne daß man sich der nervtötenden Fragerei zu Anfang noch einmal unterziehen muß. Wer besonders rationell vorgehen will, sichert (restore) seinen Spielstand vor jedem riskanten Manöver mit F7.

So verbessern sich die Chancen, im Spiel fortzuschreiten, amüsante Überraschungen und witzige Szenen zu erleben. Wer hätte gedacht, daß der unerotische Junggeselle in den heiligen Stand der Ehe tritt? Und dies nur, um mit der Angebeteten das Bett zu teilen? Allerdings reichen nicht einmal die Ketten der Ehe aus, eine Frau an Larry zu binden. Die Braut fesselt den Liebestollen ans Bett und macht sich davon.

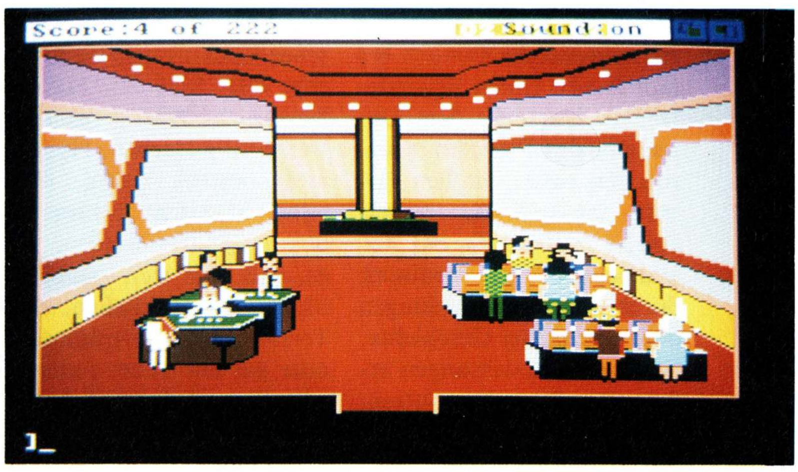
Selbst, wenn es nicht gelingt, Larry's Versuche mit Erfolg zu krönen, so erheitert doch jede Szene mit flotten Sprüchen und einfallsreichen Animationen.

(ub)

Name: Leisure Suit Larry in the Land of the Lounge Lizards

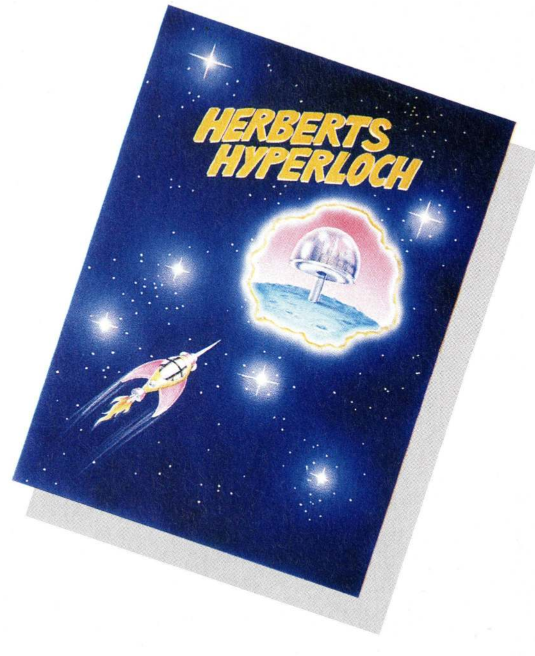
Hersteller: Sierra
Vertrieb: Ariolasoft, Leisuresoft

Spaß	10
Grafik	9
Sound	5
Idee	9
Schwierigkeit	10



Die mechanischen Slotmaschinen im Casino sind kein Ersatz für Larry's wahre Bedürfnisse.

Hundert Stunden Abenteuer



Nach vierjähriger Vorbereitung bringt die junge Gruppe „Electric Novel“ das erste ernstzunehmende deutsche Textadventure auf den Markt. Mehr noch: Mit „Herberts Hyperloch“ verschmelzen die Medien Buch und Computer zum spannenden elektronischen Roman.

Eine halbvolle Bierflasche fällt vom 10. Stock eines Wohnhauses und platscht nicht auf, statt dessen verschwindet sie in einem Hyperloch und landet auf dem Schädel eines extraterrestrischen Politikers. Dem geht es daraufhin wie einigen seiner bundesrepublikanischen Berufskollegen: Er kann sich plötzlich an nichts mehr erinnern. Electric Novels erster elektronischer Roman entführt den Spieler auf unterhaltsame Weise zu einer Reise durch Raum und Zeit. Alles beginnt ganz harmlos auf einer Straße, doch bereits nach dem ersten Spielzug geht es rund: Verseuchte Hamburger fliegen durch die Luft, eine öffentliche Telefonzelle kann sich nicht mehr beruhigen und der sternenreisende Xemock berichtet von einem bevorstehenden Unglück, dem Untergang des Planeten Xarzron. Im elektronischen Roman schlüpft der Leser/Spieler in die Rolle des Romanhelden und bestimmt durch seine Entscheidungen den Verlauf der Geschichte. Die Befehlsanweisungen erfolgen in deutschen Sätzen. Das Vokabular von „Herberts Hyperloch“ beinhaltet mehr als 2000 Worte. Der Parser, also jener Programmteil der für die Sprachanalyse zuständig ist, schluckt fast alles. Will man eine Treppe hinauf gehen sind folgende Kommandos möglich:

— Gehe die Treppe hinauf
— Ich will nach oben gehen
— Renne die Treppe rauf
— Hinauf

Sogar grammatikalisch mißglückte Eingaben kann der Parser bewältigen: „Nach ich gehen will oben die Treppe.“ Damit haben Electric Novels Programmierer ein kleines Wunder vollbracht. Besonders im Hinblick auf die bislang nur ansatzweise widerlegte Behauptung, daß ein intelligenter deutscher Parser auf Grund der komplizierten Grammatik nicht machbar sei. So konnte in der Tat kein deutsches Abenteuerspiel der englischen Konkurrenz, wie Infocom oder Magnetic Scrolls bislang das Wasser reichen. Der Witz in Herberts Hyperloch besteht darin, daß selbst idiotische Kommandos ausgeführt werden, sei es, daß man sich mit einer Mülltonne unterhalten, eine Telefonzelle zertrümmern oder gar ein Gebet einlegen will. Erfreulich ist, daß keine abwegigen Lösungen gesucht werden müssen. Intelligenz, Logik und Strategie sind die Mittel mit denen man sämtliche Abenteuer in allen vier Spielwelten bewältigen kann. Gegenwart, Vergangenheit, Zukunft und Unterwelt bestehen aus über hundert Räumen, Grafik und Sound gibt es leider nicht. Die Textmenge des Spiels entspricht dem eines 200seitigen Romans. Die Programmierung von Herberts Hyperloch hat über vier Jahre gedauert und Programmierer manch schlaflose Nacht gekostet. Die hauptsächliche Arbeit bestand in der Entwicklung eines Interpreter-Systems, das elektronische Romane möglich macht. Erstmals ist es gelungen, einen deutschen Parser zu entwickeln, der selbst komplizierte Befehlssätze verarbeiten kann. Bei jedem Spielzug in Herberts Hyperloch gibt es ungezählte Möglichkeiten. Um zu gewährleisten, daß der Parser stets intelligent antwortet, wurden sämtliche Möglichkeiten von einem Großrechner tagelang durchprobiert. Dazu kamen zahlreiche Testspieler — (einige hatten noch nie zuvor einen Computer bedient) — die in monatelanger Kleinarbeit versuchten, den Parser zu überlisten und zum Absturz zu bringen. Im Gegensatz zu herkömmlichen Parsern arbeitet Electronic Novels Parser nicht auf interpretativer Ebene. Er gaukelt also kein Verstehen vor, sondern „versteht“ tatsächlich den eingegebenen Befehl. Antworten wie „Das geht nicht!“ gibt es nicht. Wenn etwas nicht geht, dann erklärt der Parser warum. Wer gut programmieren kann, ist noch lange kein Schriftsteller, deshalb arbeiten zur Zeit mehrere deutsche Fantasie-Autoren an den nächsten Electric-Novel Romanen. In der Verpakkung von Herberts Hyperloch finden sich neben Spieldiskette und 16seitiger Anleitung witzige Beilagen, die wichtige Informationen beinhalten: Ein Zeitungsartikel aus der Bre-Tal-Grütze-Galaxie und ein Einweg-Zeitmaschinen-Reparatur-Set zum Selberbauen.

Das Spiel ist streckenweise etwas kindlich geraten, aber dennoch sowohl für erfahrene Abenteurer als auch für Einsteiger empfehlenswert. *(mss)*

Herberts Hyperloch

Preis: 89 Mark
Hersteller: Electric Novel
Görresstr. 27, 8000 München 40

Wertung	Punkte
Spaß	7
Parser	9
Idee	8
Schwierigkeit	10

Tips zu „Leisure Suit Larry in the Land of the Lounge Lizards“

Sierras Adult Software läßt die Telefone heißlaufen. Wer kennt den nächsten Schritt? Kollegen und Freunde scharen sich um den Bildschirm, um den unscheinbaren Larry endlich zu seiner Traumfrau zu führen. Wir helfen dabei ein wenig.

1 Bar:
— open door
— sit down
— order drink
— take whiskey
— stand up

2 Nebenraum:
— talk guy
— give whiskey
— take remote control
— look table
— take rose
— open door

3 Toilette:
— sit down
— use toilet paper
— look wall (mehrmals)
— wash hands
— swing hands
— look sink
— take ring
— open door

4 Straße:
— call taxi
— get in

5 Taxi:
— to shop
— pay
— get out

6 Shop:
— take magazine
— read magazine
— look wall
— ask for condome
— smooth
— colored
— rough-cut
— striped
— peppermint
— pay

7 Straße:
— call taxi
— get in
— to bar
— pay
— get out

8 Bar:
— open door
— knock door
— ken sent me
— use remote control
— change channel (mehrmals)
— look table
— take candy
— undress
— put on condome
— get in bed
— take off condom
— open window

9 Hinterhof:
— climb out
— search bin
— take hammer
— climb out

10 Straße:
— call taxi
— get in

11 Taxi:
— to casino
— pay
— get out

12 Casino:
— buy apple
— sit down
— change bet
— look ashtray
— take card

13 Straße:
— call taxi
— get in
— to disco
— pay
— get out

14 Disco:
— show card
— sit down
— look girl
— talk girl (mehrmals)
— give rose
— talk girl
— give candy
— talk girl
— give ring
— like dancing
— stand up
— sit down
— look girl
— give money
— stand up

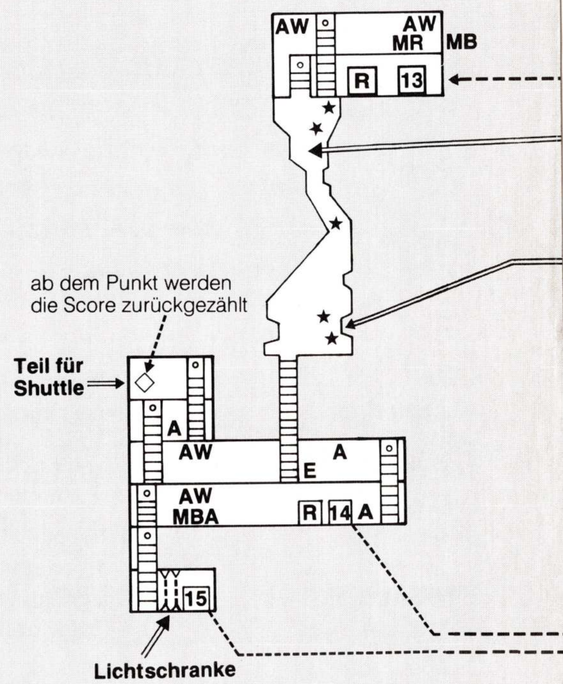
15 Straße:
— call taxi
— get in
— to chapel
— pay
— get out

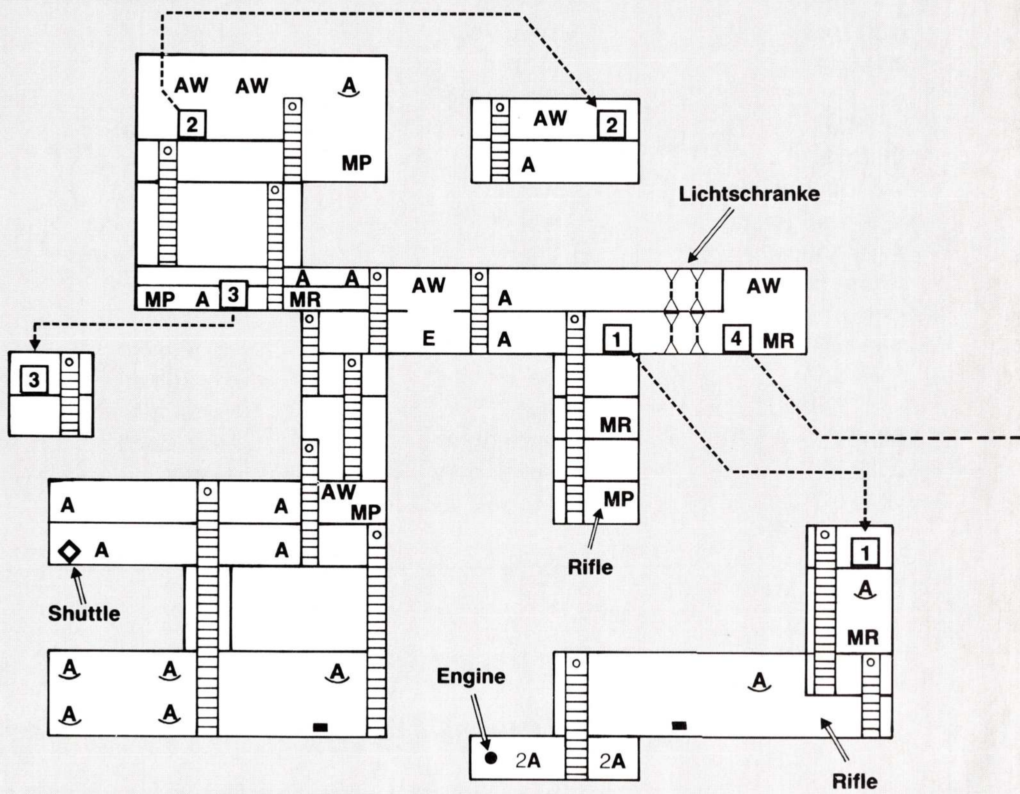
Damit ist das Adventure halb gelöst. In der nächsten Ausgabe der AmigaWelt gibt es weitere Lösungshilfen.
(Marc Biker)

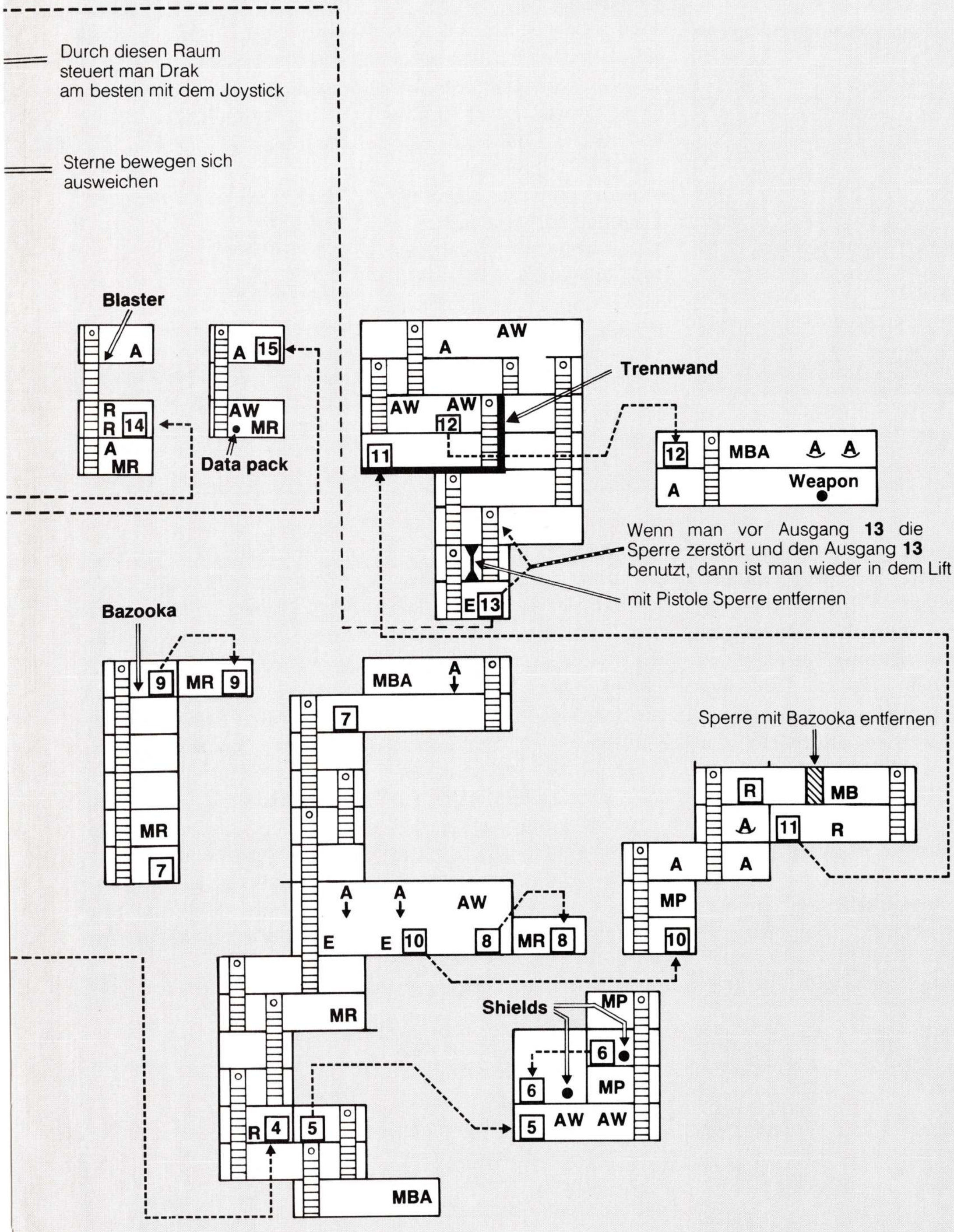
Auf der Spur von „Obliterator“

Legende:

MP = Munition Pistole
MR = Munition Rifle
MB = Munition Blaster
MBA = Munition Bazooka
AW = Automatik Waffen
A = Alien
A (mit Gleiter-Symbol) = Alien auf Gleitern
(Symbol) = Lichtschranke
A ↓ E R = lassen Eier fallen aus dem ein Alien kommt = Energiefeld = Recognizers
● Engine, Shields, Shuttle, Weapons
▬ Springen
(Symbol) mit Pistole Sperre entfernen
◆ Shuttle
[R] Personal energy shield will be repaired (Betreten mit Icon/INTO)
(Symbol) Lift

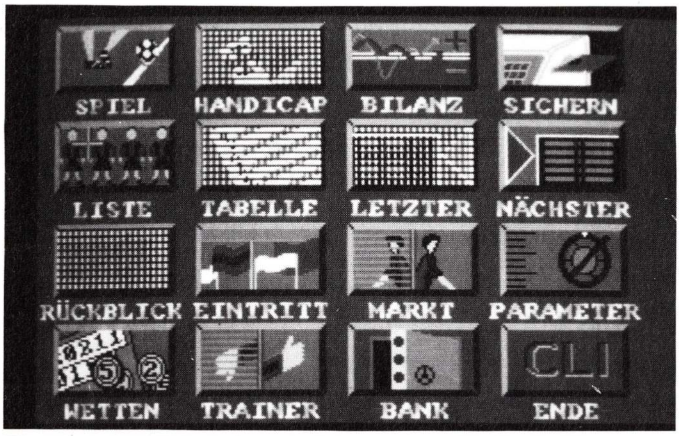





Durch diesen Raum steuert man Drak am besten mit dem Joystick
Sterne bewegen sich ausweichen
Blaster
Data pack
Trennwand
Weapon
Wenn man vor Ausgang 13 die Sperre zerstört und den Ausgang 13 benutzt, dann ist man wieder in dem Lift
mit Pistole Sperre entfernen
Bazooka
Sperre mit Bazooka entfernen
Shields

Eigentor!

„The German Football Simulation“ soll Fußballfans von der zugigen Südkurve an den heimischen Amiga locken. Hinter den Kulissen darf agiert werde. Computerspieler avancieren zum Manager und beeinflussen mit Maus und Intuition die Geschicke ihres bevorzugten Fußballvereins. Vielversprechende Spieler einkaufen, schlechte auf dem internationalen Spielermarkt abstoßen. Authentische Kickerlisten erleichtern die Wahl. Wer die Szene kennt, trifft mit etwas Gespür die richtige Entscheidung. Zusätzlich sind Spiele zu organisieren, Eintrittspreise festzulegen. Die Kasse muß stimmen — hart ist der Alltag eines Topmanagers. Soweit ist die Simulation gelungen. Allerdings haben die Programmierer nicht bedacht, daß die Begegnung interessiert. So richtig spannend ist doch erst der Wettkampf auf dem grünen Rasen. Davon jedoch sieht der Spieler überhaupt nichts. Alle Spiele laufen rechnerintern ab. Keine Grafik, keine Animation, weder Foul noch Fair. Lediglich Ergebnisse, Anzahl der Tore und Name der Torschützen, werden auf einer Anzeigetafel sichtbar. Jetzt zeigt sich, ob der Manager das optimale Team zusammengestellt, ob er die Stärken einzelner Spieler richtig eingeschätzt hat. Doch weder Tore noch Sieg können begeistern. Ladezeiten (Kopierschutz!) sind zu lang; man wartet länger als man im Endeffekt spielt. Motivation will nicht so recht aufkommen. Zwar erfreut die Titelmelodie das Ohr, aber im Spielverlauf sind weder Tröten noch Jubel und auch keine Buhrufe zu hören. All diese Mängel komplettiert der Fehlgriff beim Titel. Denn Fußball heißt im Englischen keineswegs Football, sondern Soccer. — Fußballfreunde können getrost verzichten. *(cb)*



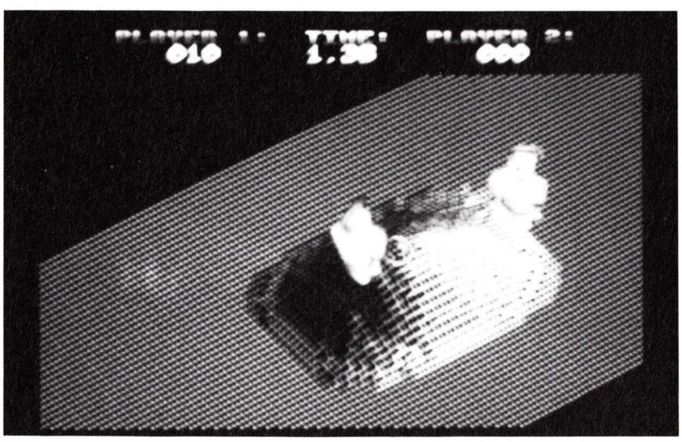
Ein übersichtliches, buntes Menü bietet Fußball-Managern alle erforderlichen Optionen.

Name: The German Football Simulation
Preis: ca. 60 Mark
Hersteller: EAS Computertechnik

Spaß	2
Grafik	3
Titelsound	9
Idee	4
Schwierigkeit	6

Kicken im Rasterfeld

Zwei goldglänzende Roboter kämpfen in der Disziplin „Vectorball“ um den Sieg. Bei dieser Mischung aus Fußball, Hockey und Rugby gibt es keinen Schiedsrichter und infolge dessen auch keine Fouls. So gelten die Attacken meist nicht dem Ball, sondern dem ballführenden Spieler. Der wird unsanft aus dem Weg geschubst, um den eigenen Spieler in Schußrichtung bringen zu können. Damit Tore und Siege nicht allzuleicht errungen werden, ändert sich das Spielfeld nach jeder Viertelzeit. Wie der Boden beschaffen sein soll, wählt man im Menü. Er kann aus Mulden und Buckeln oder aus Bergen und Tälern bestehen. Für die Darstellung klickt man vorzugsweise „Dot Pitch“ an; kurz darauf erscheint eine Vektorgrafik mit Pünktchen. Geduldige Naturen, die mindestens zwei Minuten warten können, entscheiden sich für „Shadow Pitch“, die ausgefüllte Vektorgrafik. Wenn das Programm im zweiten Viertel des Spiels dann auch noch abstürzt, ist die Motivation fast dahin. Einzelspieler im Kampf mit dem computergesteuerten, praktisch unbesiegbaren Gegner verlieren daraufhin endgültig den Spaß an „Vectorball“. Spannung kommt im Zweispielermodus auf. Hier muß der Sieger mehr als die Hälfte aller Viertelrunden gewonnen haben. Viermal drei Minuten geht es drunter und drüber, ehe ein Spiel entschieden ist. Gewöhnungsbedürftig ist das dem Ball folgende Scrolling. Der eigene Spieler kann leicht außer Sicht geraten. Wer damit vertraut ist, kann dies gezielt einsetzen und den Gegner aus dem „Nichts“ angreifen. Spannung, Grafik und Sound machen „Vectorball“ zu einem Spiel gehobener Klasse. Diesen Eindruck trübt die Tatsache, daß es sich nicht in den 2000er laden läßt und auf dem 500er nicht absturzfrei läuft.
(mms)



Goldene Roboter streiten auf einem metallenem, verbeulten Spielfeld um den Ball.

Name: Vectorball
Preis: ca. 60 Mark
Hersteller: Mastertronic
Vertrieb: Rushware

Spaß	8
Grafik	8
Sound	6
Idee	6
Schwierigkeit	7

Nordisches Heldenepos

Auf einem vergilbten Pergament ist die Vorgeschichte von „Beyond the Ice Palace" zu lesen: Einst gab es im hohen Norden ein mystisches Land, in dem Magier, Trolle, Geister und Monster lebten. Lange herrschten Eintracht und Frieden, doch wuchsen unaufhaltsam die Kräfte des Bösen. So versammelten sich die Häuptlinge des Landes und verliehen einem heiligen Pfeil die Kräfte des Waldes; in größter Not würde dieser Pfeil Rettung bringen, denn wer ihn fände, würde in einem Kampf gegen das Böse verwickelt.



Flattertiere und Trolle stürzen sich auf den edlen Mann des Nordens. Gute Geister eilen ihm zu Hilfe.

In unseren Tagen, in denen Trolle und Sagengestalten in Märchenbüchern und Verstecken verschwunden sind, findet ein Computerspieler den ominösen Pfeil. Die guten Geister des Waldes werden ihm bei seinen Abenteuern helfen und sich für das Gute opfern. Ein solches Geschenk sollte der Held jedoch sparsam einsetzen, da der gefährlichste Gegner am Ende jedes Levels wartet. Dorthin zu gelangen, ist strapaziös, denn auf dem Weg über Leitern, Treppen und Aufzüge greifen fliegende und kriechende Untiere an. Mit Schwert, Morgenstern und Messer geht es ihnen an's Leder. Doch verträgt manches Monstrum vier oder fünf Treffer, ehe es sein Leben aushaucht. Im Nahkampf ist besondere Vorsicht geboten, da Berührung mit bestimmten Wesen den Tod bedeuten kann. Wichtig ist, den Überblick zu behalten und fleißig zu schießen. Bei spannender Action und starkem Sound ist das kein Problem. Zwar wurden Amigas grafische Fähigkeiten eindeutig unterfordert, doch bietet „Beyond the Ice Palace" eine Menge Spaß für Spieler, die gern den Feuerknopf traktieren.
(mms)

Name: Beyond the Ice Palace
Preis: 60 Mark
Hersteller: Elite
Vertrieb: Rushware, Ariolasoft

Spaß	7
Grafik	4
Sound	8
Idee	5
Schwierigkeit	8

Mit Waffen Frieden schaffen?

Nur einer kann die Welt retten! Dieses Klischee wird in „Fire and Forget" nicht verlassen. 18 regionale Konflikte drohen den nuklearen Weltkrieg heraufzubeschwören. In Regierungskreisen weiß man keinen Rat mehr. Rettung könnte nur die Geheimwaffe Thunder Master bringen. Dieses Wunderwerk der Technik — salopp gesagt: ein Bastard aus Porsche und Panzer — hat schier unerschöpfliche Feuerkraft, Turbomotor, Allradantrieb und 16 Ventile. Eine Spezialkarosserie schützt Mensch und Maschine. Für seine Mission „Ballern für den Frieden" rast der Held im Thunder Master mit 265 km/h durch Feindgebiet und zerstört Hindernisse und feindliche Kriegsmaschinen. Weder Hubschrauber noch Panzer können viel gegen ihn ausrichten. Allerdings ist der Kampf nicht so einfach, wie es klingt, denn die 18 Kriege sind auf drei Level mit ansteigendem Schwierigkeitsgrad verteilt. Im Invasion Level läßt sich die Mission noch erfolgreich erfüllen, im Battle Level sind jedoch höchste Konzentration und viel Geschick gefordert. Den dritten Level zu bewältigen, ist fast unmöglich, da der Feind bereits mit zwei Treffern den Tank des Wundergefährtes leeren kann. Und ohne Kerosin ist Thunder Master aufgeschmissen; sein enormer Spritbedarf ist die Achillesferse. Unabhängig von der Fahrtgeschwindigkeit rast der Zeiger der Sprituhr beständig gegen Null, und das wird bei jedem Treffer des Feindes beschleunigt. So muß der Spieler mit Argusaugen nach den lebenswichtigen Kerosintanks ausspähen, die den Tank füllen und Bonuspunkte bringen.



Mit 265 km/h rast der mit Kanonen bewehrte Straßenflitzer durch Feindgebiet.

„Fire and Forget" ähnelt „Crazy Cars", nur daß dieses neue Spiel als Ballern mit Gegenverkehr variiert wurde. Wer Geschwindigkeitsrausch und Knallerei schätzt, ist mit „Fire an Forget" gut beraten.
(mms)

Name: Fire and Forget
Preis: 80 Mark
Hersteller: Titus
Vertrie: Rushware, GTI

Spaß	6
Grafik	7
Sound	7
Idee	3
Schwierigkeit	8

VORSCHAU

1/89 Januar/Februar

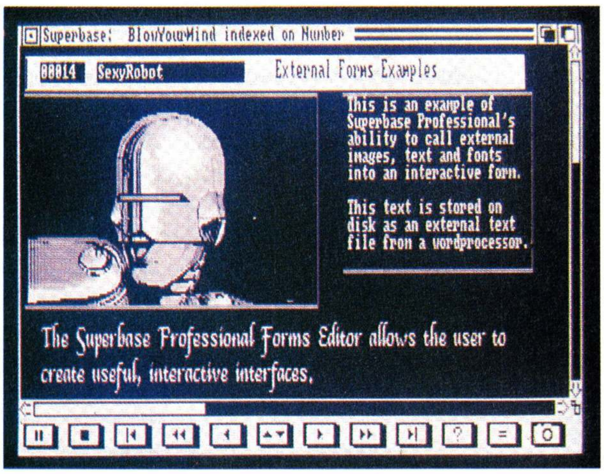


Dieser venezianische Karnevalist hilft Neulingen, die Feinheiten des Standard-Malprogramms „Deluxe Paint2“ zu erkennen und zu nutzen. Der Fortgeschrittene wird mit professioneller Software vertraut gemacht, und auch der Spielefan kommt auf seine Kosten.



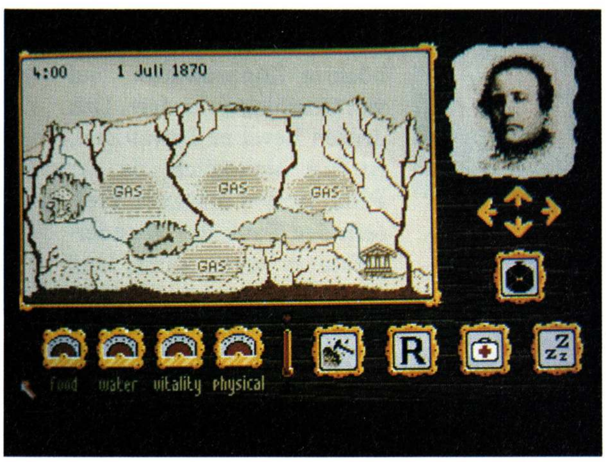
Video und kein Ende

Wem bei der Betrachtung von Videoaufnahmen beim besten Willen nicht mehr einfällt, wann die Aufnahmen gemacht worden sind und welche Verwandte und (falsche) Bekannte auf dem Bildschirm herumstolpern, sollte sich unsere schrittweise Einführung in die Technik computerisierter Texteinblendungen nicht entgehen lassen.



Amiga wird professionell

Die neueste Version der Datenverwaltung „Superbase“ nennt sich zu Recht „professional“. Sie beweist, daß mit Amiga anspruchsvolle Anwendungen realisierbar sind. Ein Kurs zeigt (an Beispielen) die Leistungsfähigkeit des Programms und erleichtert dem User den Einstieg. Ob eine Textverarbeitung existiert, die professionell einsetzbar ist, steht auf einem anderen — jedenfalls in diesem Blatt.



Eine Reise nach unten

Auf den Spuren von Arne Saknussemm geht es in eine „Reise zum Mittelpunkt der Erde“. Als Spieler begleitet Sie eine Gruppe von Abenteurern in eine unbekannte Welt. Unter der Erdkruste warten Champignonwälder, Mammuthorden und 400 KB Grafikdaten. Das Adventure enthält nicht nur reizvolle Arcadespiele, sondern ist auch eine Hommage an den großartigen Schriftsteller Jules Verne.



JETZT NEUHEITEN FEUERWERK II
links: ROBBEARY
Bertie, der Bär, ist unterwegs zu einem Einkaufsbummel im Kaufhaus. Jede Etage ist voll von verschiedenen Schätzen und Überraschungen. Tolle Grafik und Animation im besten Automaten-Stil.
Amiga 85567 59.95
rechts: HIGHWAY HAWKS
Um diesem nervenaufreibenden Autorennen gewachsen zu sein, müssen Sie nicht nur ein guter Fahrer sein, sondern auch das geeignete Zubehör für Ihren Wagen einkaufen.
Amiga 85577 59.95
STRIP POKER II PLUS DATA DISK
Lange erwartet- jetzt endlich lieferbar: Zusatzdisketten zu Strip Poker II Plus (das Originalspiel wird benötigt). Wählen Sie zwischen reizenden Damen (Beverly & Dawn) oder smarten Herren (Lee & Roy).
Disk 1 (Beverly & Dawn)
Atari ST 75617 29.95
Amiga 85617 29.95
Disk 2 (Lee & Roy)
Atari ST 75627 29.95
Amiga 85627 29.95
links: SCORPIO
Das neueste Weltraum-Actionspiel, das exzellente Grafik, Super-Digisounds und enormen Spielspaß dank zahlreicher Extrawaffen bietet. "Scorpio ist im Moment eines der besten Ballerspiele für den Amiga" (Powerplay)
C-64 Kassette 25497 29.95
C-64 Diskette 28437 34.95
Amiga 85437 59.95
links: QUANTOX
Kämpfen Sie sich durch 24 Level dieses schnellen, furiosen Actionspiels. Blitzschnelle Reaktionen sind nicht allein entscheidend, die richtige Wahl zwischen Angriff und Verteidigung ist mindestens genauso wichtig.
Amiga 85567 49.95
links: SOMMER OLYMPIADE
Unser berühmtes Sportspiel, das mit dem goldenen Software-Oskar ausgezeichnet wurde, gibt es jetzt in einer tollen Umsetzung mit Spitzen-Grafik und -Animation auch für den Amiga.
C-16, Plus/4 Kassette 35207 29.95
C-16, Plus/4 Diskette 38207 29.95
Amiga 85207 59.95
links: PLATOU
Die einfachsten Spielideen sind oft die besten - zwei Kugeln auf dem Weg zu ihren Ziel-Plattformen. 1 oder 2 Spieler (gegeneinander) können an diesem neuartigen Spaß teilnehmen, ein Editor sorgt für viele neue Level.
C-64 Kassette 25397 14.95
C-64 Diskette 28397 19.95
Amiga 85397 39.95
rechts: FACE OFF
Erleben Sie das atemberaubende Tempo und den enormen Spielspaß dieser perfekten Eishockey-Simulation. Natürlich mit Bodychecks und Fouls, Ligamodus und 1/2-Spieler-Option.
Atari ST 75597 49.95
Amiga 85597 49.95
WANTED KINGSOFT sucht PROGRAMMIERER
für fast alle Computer-Typen, die gegen erstklassige Bezahlung Spiele von internationalem Niveau schreiben. Wenn Sie Interesse haben, wenden Sie sich noch heute an uns.
SPITZEN-SOFTWARE
KINGSOFT
MADE IN GERMANY
Grüner Weg 29
5100 Aachen
Tel. 0241/15 20 51
Fax 0241/15 20 54
Dies ist nur ein kleiner Ausschnitt aus unserem riesigen Softwareangebot. Fordern Sie bitte noch heute kostenlose Informationen an.
Alle Spiele selbstverständlich mit ausführlicher deutscher Anleitung. Bitte bei Ihrer Bestellung unbedingt die genaue Bestell.-Nr. angeben.